UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

APLICACIÓN WEB PARA LA ENSEÑANZA DE C

Felipe Millán Fernández
Tutor: Alejandro Sierra Urrecho

JULIO 2015

APLICACIÓN WEB PARA LA ENSEÑANZA DE C

AUTOR: Felipe Millán Fernández
TUTOR: Alejandro Sierra Urrecho

Dpto. Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Julio de 2015

Resumen

El objetivo de este TFG es el desarrollo de una aplicación web para el aprendizaje autónomo del lenguaje de programación C. La aplicación proporciona información detallada al profesor sobre el proceso de aprendizaje, permitiendo así mejorar el contenido del curso entre ediciones sucesivas. La aplicación podría llegar a ser de mucha utilidad para las asignaturas de programación de los grados de la Escuela Politécnica Superior de la UAM. Además, se puede adaptar con facilidad a otros lenguajes de programación como Java o Android.

Existen muchos sitios web de aprendizaje autónomo para lenguajes naturales como el inglés, pero en el caso de los lenguajes de programación, estos son relativamente escasos. Actualmente, han surgido nuevas iniciativas que están teniendo mucho éxito como codecademy o codeschool. La mayoría ofrecen cursos de lenguajes de marcado como HTML o lenguajes interpretados como JavaScript, que permiten al estudiante probar su código rápidamente y sin instalar ningún entorno de desarrollo. En este trabajo, se extiende esta idea a los lenguajes compilados como C.

Uno de los objetivos del trabajo, además del educativo, ha sido el desarrollo desde cero de una aplicación web sencilla pero profesional. Se ha hecho un esfuerzo didáctico para que este documento sirva para futuros desarrollos web similares. Por un lado, se explican las distintas actividades de desarrollo de un producto software, haciendo uso de los conocimientos adquiridos a lo largo de la carrera. Por otro lado, se aplican estos conocimientos en un entorno de desarrollo en concreto, llamado Laravel, un framework para implementar código PHP de forma simple y elegante, con una gran influencia de otros como Ruby on Rails y ASP .NET.

Palabras clave

Educación, ingeniería del software, aplicación web, lenguaje de programación, Laravel

Abstract

The objective of this Bachelor Thesis is to develop a web application for the autonomous learning of the C programming language. The application gives detailed information to the teacher about the learning process, so as to improve the course's contents between successive editions. The application could be very useful for the programming subjects of the degrees of the Escuela Politécnica Superior of the UAM. Moreover, it can be easily adapted to other programming languages such as Java or Android.

There exist many web sites for the autonomous learning of natural languages such as English, but with respect to programming languages, these sites are relatively few. Lately, some new initiatives such as codecademy and codeschool have emerged. Most of them offer programming courses for markup languages such as HTML or interpreted programming languages such as JavaScript, which allow the student to check her code very quickly and without the setting of any development environment. In this work, this idea is applied to compiled languages such as C.

One of the objectives of this work, apart from the educational, is the development from scratch of a simple but professional web application. A considerable effort has been made for this document to help in future similar developments. On the one hand, the various phases in the development of a software product have been explained, making use of the knowledge acquired during the degree. On the other hand, this knowledge has been applied to a concrete development environment, called Laravel, an elegant and simple PHP framework, greatly influenced by Ruby on Rails and ASP .NET.

Index terms

Education, software engineering, web application, programming language, Laravel.

Agradecimientos

En primer lugar, agradecer a mi tutor Alejandro por el tiempo que ha dedicado en este trabajo y la paciencia que ha tenido. Por otra parte, dar las gracias a todos los profesores que han formado parte de la enseñanza durante todos estos largos años de la carrera, así como a todos los compañeros con los que he compartido horas de clase y de prácticas. Sin ellos no hubiera aprendido tantas otras cosas que no te dan un libro o una transparencia de teoría y hace que se lleve de otra forma los exámenes. También quiero mencionar al resto de alumnos, que de forma desinteresada, han estado ahí cuando ha hecho falta.

Agradecer a la universidad Autónoma de Madrid y en particular a todo el personal de la Escuela Politécnica Superior, trabajadores, Biblioteca, Administración, etc., por resolver todas las dudas e informar siempre correctamente.

Tampoco quiero olvidarme de mis padres, por toda la ayuda que he recibido de ellos. Y por último, gracias especialmente a mi novia Mª Carmen, a quien dedico este trabajo final, por todo el apoyo recibido y que me ha permitido llegar hasta aquí, superando todos los momentos difíciles que he tenido. Se cierra una etapa en mi vida, pero se abre otra a tu lado, llena de nuevas ilusiones y retos profesionales.

Índice de contenido

Índice de tablas

Índice de figuras

1. Introducción

En la actualidad, las aplicaciones web aportan una serie de utilidades renovadas al usuario y aunque parezca sorprendente, debido al continuo avance y desarrollo de las tecnologías de la información y comunicación, aparecen nuevas necesidades que satisfacer. Esta capacidad de adaptación a la demanda que se produce en cada momento, es lo que hace distinguir a estas tecnologías del resto. Este trabajo de fin de grado forma parte del área de la ingeniería del software y de las tecnologías de la información y computación (departamento de Ingeniería Informática) y se van a exponer las ideas para la creación de una aplicación web con una funcionalidad concreta en dicho área.

1.1. Motivación

Las aplicaciones para la enseñanza de un curso básico o más avanzado de cualquier tipo de tema siempre ha supuesto un punto de motivación especial, en el que se intenta ayudar desde a una organización hasta a cualquier usuario en particular. Actualmente, se dispone de numerosos sitios sobre el aprendizaje automático. Lo que parecía una novedad se ha convertido en estos últimos años en una realidad: a la sociedad le gusta aprender rápida y cómodamente desde cualquier sitio y con el uso de las redes sociales, compartir su experiencia con los demás. Es un hecho que los sitios estáticos desaparecen y no tienen las grandes ventajas con las que cuentan aplicaciones que se conectan a Internet.

Pero ofrecer un sistema de aprendizaje electrónico sobre un curso de cocina por ejemplo, o un curso de inglés básico ha dejado de ser especialmente llamativo, ya que existen muchos recursos para ello, a través de vídeo tutoriales, video llamadas, artículos, etc. Con relación al área de este proyecto, también existen muchos entornos de aprendizaje autónomo que hacen uso de esta tecnología, pero llama la atención que en el caso de los lenguajes de programación, éstos son relativamente escasos.

La creación de un sitio web es parte de la competencia actual y que esté basado en la educación no le suma un mayor protagonismo, al contar con numerosas aplicaciones de propósito administrativo, docente y de investigación. Lo que distingue a unas de otras es la capacidad de innovación a la hora de crear los contenidos. La principal motivación de este proyecto reside en que los cursos de dichas aplicaciones existentes, no sirven para lenguajes de programación compilados, un requisito que será fundamental en esta aplicación.

Pero no es la única motivación. La realización de esta aplicación conlleva motivaciones secundarias, como su uso y colaboración con los primeros cursos de la Escuela Politécnica Superior (Universidad Autónoma de Madrid), que hace un proyecto todavía más atractivo.

1.2. Objetivos

Tras una primera impresión de los párrafos anteriores, se deduce la necesidad de crear un entorno web, sencillo, para el aprendizaje de un lenguaje de programación, que acompañe a un estudiante durante el primer curso de un grado en la Escuela Politécnica Superior.

Esto implica nuevos objetivos básicos pero no por ello menos importantes. El primero de ellos será el de gestionar un sistema que almacene usuarios y cursos relacionados entre ellos. El segundo, exige que haya una gran seguridad de los datos almacenados y de la funcionalidad interna de dicho sistema. Y todo ello que sea de una forma interactiva, fácil y amigable entre el usuario y la interfaz de la aplicación.

La forma de afrontar estas necesidades será clave para superar este reto. Por ello se dividirá la aplicación en varios módulos para mejorar la planificación del proyecto. El objetivo final es captar la atención del alumno en el uso de estas aplicaciones, que ayudan y mejoran la inicialización a un nuevo curso en el cuál apenas se tiene la experiencia suficiente y la desorientación y desesperación puede provocar un cierto desinterés, acabando en algunos casos más extremos, en el abandono del centro.

Una de las grandes utilidades reside en un seguimiento y control detallado del estudiante, de donde se podrá sacar datos interesantes sobre el progreso del curso y obtener estadísticas más potentes, como por ejemplo, los contenidos que han requerido mayor esfuerzo. Aquí es donde tiene lugar otro de los objetivos más importantes de la aplicación. Se deberá tener en cuenta estas opciones para permitir obtener datos que midan la calidad del curso, su popularidad o la dificultad de los distintos contenidos de cada curso.

De forma paralela, en este documento se explica todo el proceso de desarrollo de una aplicación web desde cero, aplicando todos los conocimientos adquiridos durante estos años en un *framework* sencillo y divertido, con el objetivo de seguir un orden profesional y una sintaxis común para todos sus desarrolladores posteriores y su mantenimiento.

1.3. Estructura del documento

La estructura de este documento sigue la normativa del Trabajo Final de Grado: introducción, estado del arte, análisis y diseño de la aplicación, implementación, pruebas y conclusiones.

Antes de comenzar a hablar sobre el producto, se analizan las herramientas que existen en la actualidad, para tener una idea de lo que se puede llegar a conseguir y así reforzar la base de la aplicación. Es de gran utilidad comparar las soluciones software ya disponibles para abordar los objetivos planteados, destacando tanto sus ventajas como sus inconvenientes, para luego dar paso a las tecnologías que se van a utilizar. Se realizará un breve resumen de los conocimientos necesarios para desarrollar con dichas tecnologías.

En el tercer capítulo se realiza una descripción detallada de la solución implementada, aportando el catálogo con las especificaciones de la aplicación. A continuación, en el siguiente capítulo, una vez definidos todos los requisitos necesarios, se procede a diseñar la arquitectura de la aplicación. Tener claro lo que se quiere hacer y elegir correctamente las herramientas y el lenguaje de programación, hace que sea una fase crítica del proyecto. En este apartado también se incluye diversos diagramas sobre el diseño del sistema.

Una vez explicada la funcionalidad y diseño web, en el capítulo 5 se explica la implementación de cada uno de los módulos en los que se divide la aplicación, a través del *framework* elegido. La intención es que sirva de manual para resolver los problemas básicos, programando algunos de sus algoritmos y explicando toda su lógica y las consultas a la base de datos para almacenar o recuperar los datos necesarios, así como el diseño de las plantillas web para mostrarse al usuario final.

En el capítulo 6 se realizan las pruebas de la aplicación para comprobar la eficiencia y la calidad del software producido y se analizan los resultados obtenidos. En el siguiente capítulo se enumeran todas las conclusiones, así como las nuevas ideas para futuras versiones.

Por último, en la parte final del documento se presentan los manuales de instalación y configuración necesarios para disponer del entorno completo de desarrollo, así como más información a nivel de usuario de la aplicación.

2. Estado del arte

Para cumplir los objetivos establecidos, previamente hay que comparar los sistemas similares que existen en la actualidad. En la segunda parte del capítulo, se explica la tecnología que sigue una aplicación web y los conceptos básicos que son necesarios.

2.1. Servicios actuales

Como se ha comentado en el capítulo introductorio, hoy en día existen diferentes y numerosos tipos de recursos para el aprendizaje autónomo: plataformas de docencia online que ofrecen MOOCs (edX, coursera, udacity, etc.); sitios web especializados en el aprendizaje autónomo de lenguajes de programación (codecademy, w3school, codeschool, etc.); plataformas con contenido audiovisual, actividades y seguimiento de los estudiantes matriculados (Moodle, Open edX); foros y comunidades donde se resuelven dudas y se comparten contenidos, etc.

Los sistemas que más nos interesan en este trabajo y que tienen las mejores críticas de los consumidores, son claramente dos: w3school y codecademy. Por ello han sido los elegidos en el análisis competitivo que se adjunta en la "Tabla 1". Para su estudio, se han comparado las principales características que debe requerir un sistema web y que se explica a continuación.

Los cursos de lenguajes de marcado (HTML, CSS) han tenido una gran aceptación, debido a la rapidez de uso e interpretación del código. Pero sorprende como en cuanto a los lenguajes de programación, en los que se requiere compilar, la situación no es tan favorable. El primer objetivo consiste en reunir todas las características útiles de estos cursos actuales y aplicarlas a los cursos de lenguajes de programación. En una primera aproximación, se trabajará sobre el curso C que es el impartido en el primer curso de Grado en Ingeniería Informática de EPS.

Evaluar un sitio Web puede ser bastante complejo, por lo que nos limitamos a destacar los aspectos más positivos y negativos, relacionados con una aplicación educativa o de aprendizaje [1], destinados a usuarios finales como alumnos de un curso o estudiantes en general. No existen obligaciones de ningún tipo a la hora de construir una página Web, pero sí hay una serie de recomendaciones para hacer un buen uso y estandarización. De ellos se encarga las organizaciones internacionales como W3C (World Wide Web Consortium) y herramientas que evalúan la usabilidad y accesibilidad, como TAW (pautas de accesibilidad al contenido de una aplicación Web).

	w3schools.com	codecademy
Facilidad de uso	Fácil. Menú para poder elegir en todo momento el curso.	Muy fácil. Sistema de ayuda completo.
Entorno audiovisual	Más cursos y peor organizado.	Diseño claro y muy atractivo.
Calidad del contenido	Muy bueno, con muchos cursos y contenido bien expresado. Tutoriales muy completos.	Bueno, pero con ejemplos muy básicos y no tan didácticos.
Navegación e interacción	Buena. Demasiados enlaces entre los contenidos del curso, que puede perjudicar el seguimiento.	Muy buena, con pantallas múltiples que ayuda en el aprendizaje del usuario.
Adecuación a los usuarios	Excelente. Completamente abierto. Se adapta muy bien al responsivo. Contiene buscadores de ayuda.	Excelente. Algunos cursos requieren estar identificado.
Capacidad de motivación	Sencillo. Está orientado para aprender con total libertad.	Bueno. Muestra el progreso completado de los cursos.

Tabla 1. Análisis competitivo (w3schools vs codecademy)

Como se puede comprobar, la página 'w3schools' cuenta con un contenido de aprendizaje más completo y con un buen diseño en general, pero en ocasiones ciertas características del diseño de una página web pueden ser tan importantes como la calidad del contenido, para ser completamente accesible para el usuario final de la aplicación [2]. Prueba de ello es que la página 'codecademy' cumple con todas estas recomendaciones y puede llegar a ser más conocida o visitada que la anterior.

La era de la comunicación móvil se consolida cada vez con más fuerza y es necesario adaptarse a estos cambios, provocados también por las grandes empresas, como en su día hicieron Facebook o Apple (eliminando el uso de la tecnología Flash) y recientemente Google con una nueva exigencia para los sitios Web [3], penalizando a los sitios que no sean compatibles con dispositivos móviles, es decir, que no cuenten con un diseño adaptable.

2.2. Tecnologías utilizadas

En los últimos años, ha surgido la necesidad de que la información sea accesible tanto desde cualquier lugar dentro de la organización (o red local) como desde el exterior, y que dicha información sea compartida entre todas las partes (organización y consumidores finales) en cada momento. Estas necesidades han provocado un cambio en las aplicaciones: de las tradicionales de escritorio hacia las **aplicaciones Web**. Ahora no solo se muestra información, sino que, este aumento considerable, exige otras cualidades indispensables: seguridad, escalabilidad, portabilidad, disponibilidad, eficiencia, etc. A continuación se resume alguna de estas características.

Una aplicación Web es una aplicación informática que se ejecuta en un entorno Web y permite a múltiples clientes establecer una comunicación a través de Internet con un servidor. Lo primero a tener en cuenta es el **sitio Web** donde se almacena el contenido de la aplicación (archivos, vídeos, música, imágenes y otros datos de configuración). El entorno puede ser, según su funcionamiento, una intranet o extranet. En el primer caso, la aplicación sólo estaría disponible dentro de la red interna de la organización; en el segundo, en cualquier punto con acceso a Internet. Se puede renunciar a las ventajas de estas tecnologías, o por el contrario, colocar la aplicación en un servidor con buena conexión, para llegar a usuarios que estén fuera de la red local.

El siguiente paso es contratar un **dominio Web** a una compañía especializada (registrador de dominios) a cambio de tener un nombre de dominio que sea representativo del sitio Web. El DNS Server se encarga de traducir el nombre de la URL en una dirección IP y enviar la petición web al servidor. Esta abstracción hace posible que cualquier servicio de red pueda estar disponible desde cualquier computadora en la red Internet, aun cuando la dirección IP sea diferente. Normalmente, la empresa sólo vende el nombre de dominio y proporciona el servidor DNS, pero también puede ofrecer otros servicios que se ven a continuación.

Una vez que se ha pagado por un nombre de dominio, es necesario disponer del hosting o **alojamiento Web**, es decir, de un servicio (servidor Web) donde guardar el sitio Web en Internet y que se encargue de procesar las peticiones en el puerto correspondiente del servidor. Los clientes, mediante un navegador, emiten solicitudes vía protocolo HTTP a un servidor Web, que aloja una serie de aplicaciones o páginas Web que ofrecen diferentes servicios a los usuarios conectados. Por ello, es aquí donde se centraliza todo el trabajo, mientras que en el navegador sólo se presenta la información (modelo de cliente ligero).

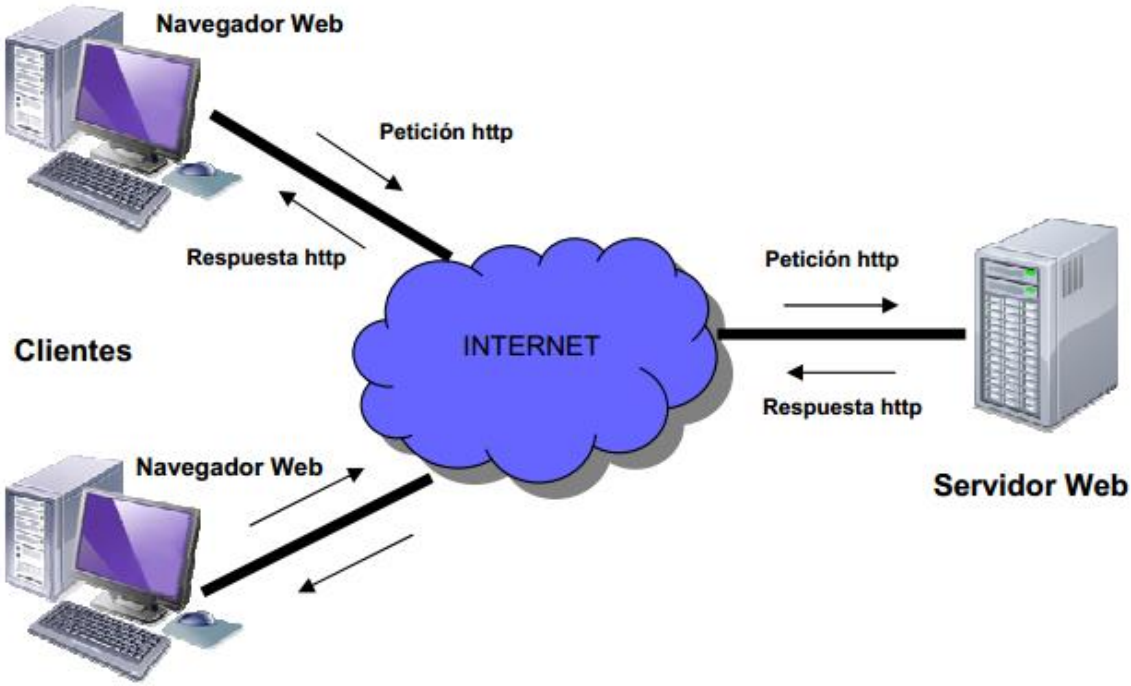

Figura 1. Arquitectura Web

Existen distintos tipos de alojamiento web en Internet, según las necesidades específicas del usuario. Entre los principales tipos se encuentran el alojamiento gratuito, que cuenta con recursos bastante limitados; el alojamiento compartido, donde se alojan clientes de varios sitios en un mismo servidor y es un servicio económico y con buen rendimiento; el alojamiento de imágenes, generalmente gratuito; servidores virtuales, recomendado para empresas de diseño y programación web; servidores dedicados, donde un solo cliente dispone de todos los recursos de la máquina pero que suele tener un coste mayor al del alojamiento compartido debido a la necesidad de contratar servicios adicionales para la administración y configuración del servidor; alojamiento web en la nube (*cloud hosting*), basado en las tecnologías más innovadoras que aportan mayor seguridad [4].

El modelo de comunicación entre cliente y servidor se establece mediante una conexión a Internet y se usa el protocolo HTTP para el intercambio de información. Para más información sobre los protocolos, se puede consultar [5]. El navegador Web interpreta las páginas dinámicas generadas por el servidor Web.

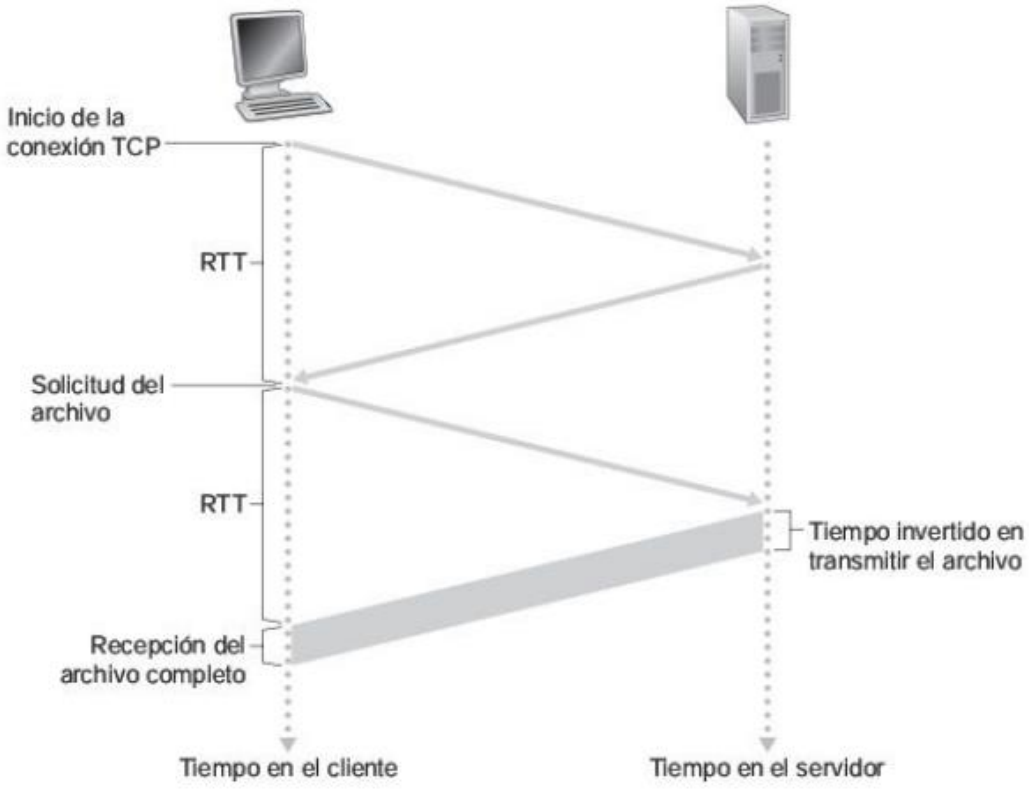

Figura 2. Comunicación HTTP

La aplicación normalmente hace uso de una base de datos para almacenar y traer toda la información necesaria. En resumen, el sistema se distribuye en tres componentes principales: el navegador, la aplicación y la base de datos. Esta estrategia hace que las aplicaciones se encuentren con importantes problemas de escalabilidad, disponibilidad, seguridad, usabilidad, eficiencia o integración, entre otros. Como solución, se ha generalizado la división de las aplicaciones Web en tres o más capas (multinivel, ver Figura 3):

- **Capa de presentación** (capa cliente), que genera la interfaz de usuario, envía los datos a la capa intermedia y procesa los resultados que llegan de ésta. La información se envía a través de formularios Web que rellena el cliente.

- **Capa de negocio** (capa intermedia), que se corresponde con el núcleo de la aplicación y contiene toda la lógica que modela los procesos de negocio.

- **Capa de datos**, que se encarga de hacer persistente toda la información y de almacenar y suministrar los datos al nivel de negocio.

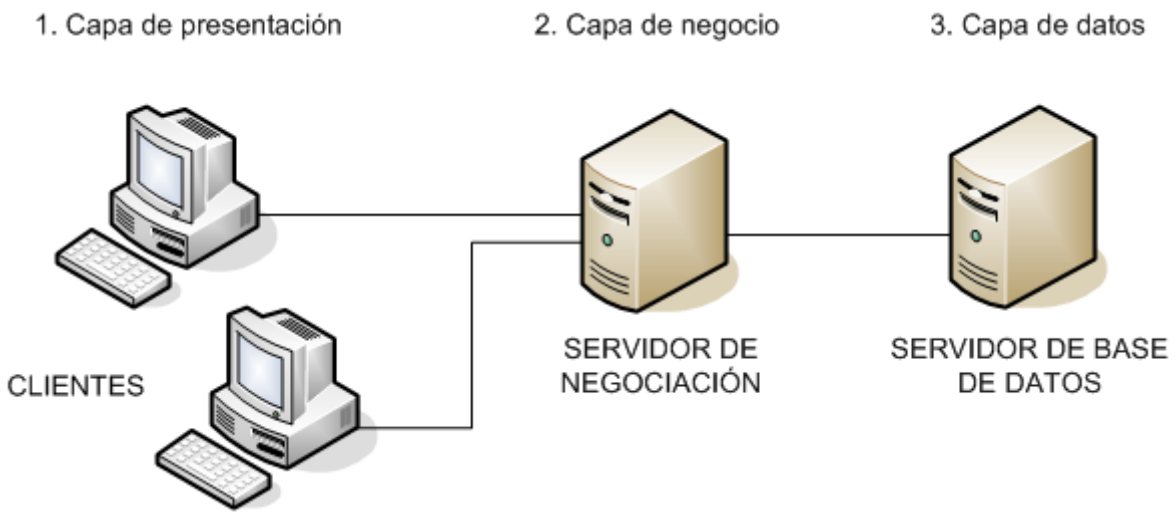

Figura 3. Arquitectura de tres capas

Cada capa interactúa solamente con sus capas adyacentes, por ejemplo, la capa de presentación no se da cuenta del tipo de base de datos utilizado. Esta separación en capas permite una gran flexibilidad a la hora de construir aplicaciones. Para una mayor facilidad de desarrollo, surge una serie de pilas de aplicaciones. La Tabla 2 muestra una lista de las más conocidas y actuales.

Servidor Web	**Framework**	**Lenguaje programación**	**Gestor base de datos**
Apache	Symfony, Laravel	PHP	PostgreSQL(phpPgAdmin)
Tomcat, Glassfish	JSP	Java	MySQL (phpMyAdmin)
IIS / IIS Express	ASP.net	VB, C#	SQL Server (Access)
NGINX	Django	Python	MySQL (phpMyAdmin)
Node.js	Express	JavaScript	MongoDB (mongoVue)

Tabla 2. Stacks más conocidos

Como se ha visto, existen numerosos *stacks* conocidos, como por ejemplo: Ruby on rails, LAMP (Linux/Apache/MySQL/PHP), MEAN (MongoDB/Express/Angular/NodeJS), etc., que facilitan el desarrollo de aplicaciones Web. Además, este tipo de arquitecturas tienen las siguientes ventajas: se consiguen aplicaciones robustas; proporcionan máxima flexibilidad al modificar módulos de una capa sin afectar a las restantes; posibilidad de reutilización de código en otras aplicaciones o versiones futuras; alta escalabilidad, añadiendo más servidores para manejar más peticiones de usuario mejorando el rendimiento.

Por el contrario, las aplicaciones Web también traen consigo varios aspectos negativos, entre los que hay que tener en cuenta: mayor incremento del tráfico en la red; mayor balance de carga y tolerancia frente a posibles fallos; mayor exigencia de seguridad ya que las aplicaciones están expuestas a ataques maliciosos.

El último paso a tener en cuenta, es el **diseño Web**. Hasta ahora, el trabajo realizado en la parte del servidor, lo realiza el programador Web o Back-End Developer. Pero la parte en la que se indica cómo se debe mostrar la información al usuario, no deja de ser menos importante. Por ello existen varias tecnologías, lenguajes como HTML5, CSS3, JavaScript y *frameworks* como por ejemplo JQuery, Bootstrap o AngularJS, de los que se encarga el diseñador Web o Front-End Developer.

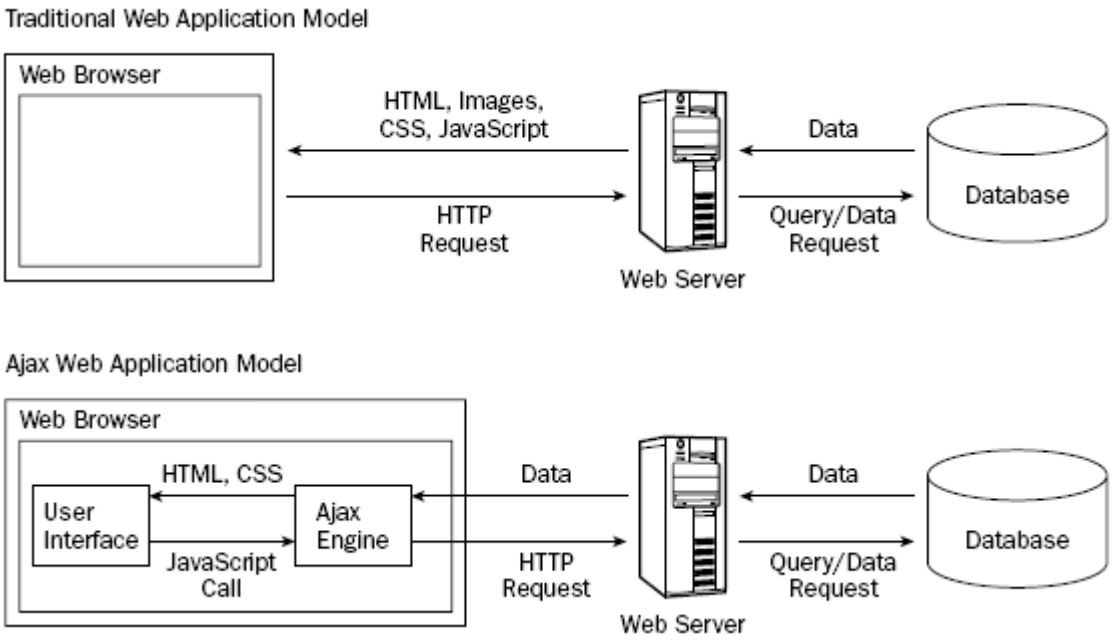

Figura 4. Tecnología AJAX

Lo cierto es que en los últimos años, el diseño Web está teniendo más importancia junto con el desarrollo del lado del servidor, y cada vez es más frecuente conectarse a los datos desde el navegador sin pasar por el servidor. Aparte de las tecnologías GET y PUT, existen otras como AJAX (Figura 4) o el uso de sockets, que resultan bastante útiles para conectarse con la base de datos en un proceso de verificación de formularios, por ejemplo. Esto es un proceso asíncrono donde se puede usar JSON o XML como forma de empaquetar la información en el intercambio de los datos.

Una de las soluciones a la hora de tratar algunos problemas complejos de la arquitectura web es seguir el patrón MVC (Modelo Vista Controlador) mostrado en la Figura 5, que se encarga de separar la lógica de negocio de la interfaz de usuario. Es uno de los más actualizados en las aplicaciones ya que facilita la funcionalidad, mantenibilidad y escalabilidad del sistema de forma sencilla.

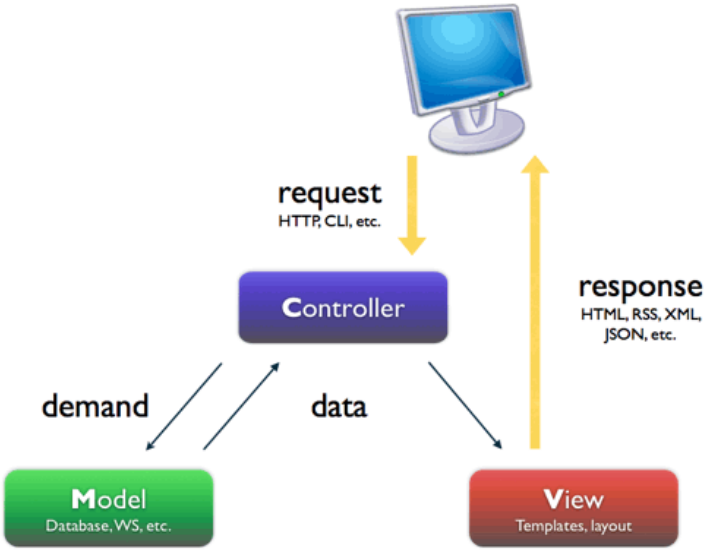

Figura 5. Arquitectura MVC

Según este patrón, la capa intermedia de una aplicación Web se divide en tres componentes: el usuario realiza una petición a través de su navegador web, el **controlador** captura el evento y determina la acción a realizar, llamando al modelo de datos en caso de ser necesario; el controlador recibe la información del **modelo** y la envía a la vista; la vista se encarga de presentar los datos recibidos del controlador en una plantilla HTML.

3. Análisis de los requisitos

En este capítulo se define el proyecto y los requisitos necesarios para su implementación. En los próximos apartados se profundiza en las características concretas del producto, así como en el catálogo de especificaciones y se expone el modelo de ciclo de vida a seguir.

3.1. Definición del proyecto

El objetivo principal de este Trabajo Fin de Grado en Ingeniería Informática es la construcción y el desarrollo de un producto software, en este caso una aplicación Web para el aprendizaje autónomo del lenguaje de programación C.

Se seguirán todas las etapas del proceso de desarrollo de software (Figura 6): planificación, análisis, diseño, implementación, pruebas, instalación y mantenimiento, usando un modelo de ciclo de vida en cascada. Es un modelo clásico, ideal para proyectos estables y pequeños, donde los requisitos quedan bien documentados al inicio del ciclo..

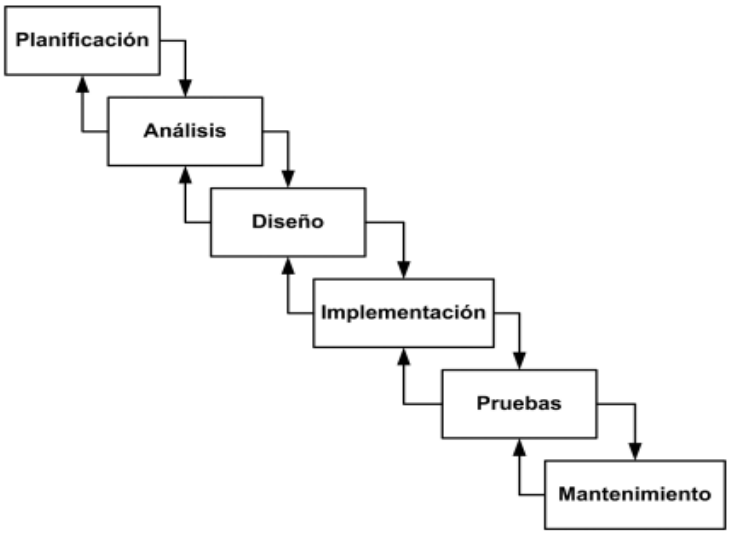

Figura 6. Ciclo de vida clásico: modelo en cascada

Cada fase se completa en orden y tras la verificación, se puede volver a la fase anterior en caso necesario. No se realizará la entrega del producto final hasta la validación de la última etapa. Esta secuencia lineal puede tener varios inconvenientes, como los riesgos de modificar los requisitos en una fase más avanzada, por ello en proyectos reales, se opta por seguir otros modelos como el iterativo, incremental, en espiral o de prototipos. Ver más en [6] y [7].

A partir de este momento en este documento, se utilizará Codeps como nombre de la aplicación real que se describe en este documento. Codeps surge de la necesidad de desarrollar un sistema Web para el aprendizaje autónomo, con los objetivos descritos en el capítulo anterior.

Figura 7. Logo de la aplicación

Se hace un guiño a la Escuela Politécnica Superior de la Universidad Autónoma de Madrid y se forma el juego de palabras para simplificar la idea general del proyecto: Code + EPS = código fácil para aprender en los cursos de la escuela EPS.

3.2. Catálogo de requisitos

Una vez que se han introducido los objetivos de la aplicación y se ha estudiado las tecnologías y servicios disponibles actualmente, es hora de exponer todos los requerimientos necesarios tanto para el usuario como para el entorno del sistema.

3.2.1. Requisitos funcionales

Aplicación

RF (1).- La aplicación no permite mostrar el contenido si el usuario no está registrado., así como participar y modificar otras configuraciones de la página.

RF (2).- El sistema dispone de un módulo de gestión de los usuarios.

RF (3).- El sistema dispone de un módulo de gestión de los contenidos del curso.

RF (4).- La aplicación acepta solamente dos tipos de usuarios: administración y estudiantes.

RF (5).- La aplicación tiene la posibilidad de vincularse con redes sociales.

RF (6).- La aplicación tiene que listar los usuarios que están registrados y todos los cursos que han sido almacenados.

RF (7).- La aplicación tiene un servicio de contacto para recibir mensajes de los usuarios a través de un correo electrónico para enviar notificaciones o sugerencias.

RF (8).- La aplicación debe permitir inscribirse en un curso, mostrando los grupos disponibles y la información en detalle de dicho curso.

RF (9).- La inscripción o registro debe almacenar campos de tipo fecha para saber cuándo se inicia y se acaba el curso.

RF (10).- La aplicación permite cargar cursos personalizados de forma automática, a través de un fichero que debe tener un formato específico, que cuente con los campos necesarios del curso (título, descripción, temas, contenidos, etc.).

RF (11).- Los cursos de la aplicación deben especificar un atributo que indique el lenguaje para que en el editor sea posible su identificación.

RF (12).- Los cursos tienen que permitir guardar al menos los estudiantes que se han inscrito, el nivel de dificultad y la duración estimada.

RF (13).- El sistema tiene que ser capaz de poder valorar un curso mediante la intervención de los alumnos registrados.

RF (14).- La aplicación permite compilar un código escrito por el usuario en tiempo real y se mostrará el resultado por pantalla.

Administrador

RF (15).- El administrador tiene acceso a secciones privadas de la aplicación, como la edición o el listado de los usuarios que hay registrados en el sistema.

RF (16).- El administrador puede acceder a las configuraciones del sistema, como la carga o edición de los contenidos de un curso y puede ver sus estadísticas.

Usuario

RF (17).- Los usuarios tienen que *registrarse* en la aplicación antes de poder ver el contenido o iniciar un curso pero pueden ver la información de los cursos que hay disponibles.

RF (18).- Los usuarios pueden *inscribirse* en un curso, seleccionando el grupo adecuado y pueden comenzar a *realizar* dicho curso, siempre que hayan iniciado la sesión.

RF (19).- Los alumnos tienen que seguir un orden en el curso, es decir, hasta que no se haya superado un contenido, no se puede realizar el siguiente.

RF (20).- Los usuarios pueden compilar el propio código que hayan escrito en el curso.

RF (21).- Los usuarios tienen que estar inscritos en un curso antes de poder usar su contenido.

3.2.2. Requisitos no funcionales

Interfaz y usabilidad

RNF (1).- La interfaz gráfica será sencilla, atractiva y fácil de manejar, basado en usuarios con unos conocimientos mínimos en Internet.

Documentación

RNF (2).- La aplicación será multilenguaje, con al menos dos idiomas: español e inglés.

RNF (3).- La aplicación tiene que disponer de una sección de ayuda, donde se oriente al usuario sobre el contenido de la aplicación.

Mantenimiento y portabilidad

RNF (4).- El administrador tendrá la opción de poder mantener el sistema con actualizaciones de versiones con el fin de corregir errores y mejorar los servicios.

RNF (5).- La aplicación será privada, permitiendo al administrador el acceso al servidor.

Recursos

RNF (6).- La información de todos los usuarios de la aplicación se almacenará en una base de datos y se dispondrá de un módulo de gestión de alumnos y administradores.

RNF (7).- La información de todos los cursos de la aplicación se almacenará en una base de datos y se dispondrá de un módulo de gestión de cursos y contenidos.

Verificación y fiabilidad

RNF (8).- Para registrar un usuario, habrá que realizar un cuestionario con información obligatoria como el nombre de usuario, contraseña y correo electrónico.

Rendimiento

RNF (9).- La aplicación permitirá el acceso a un usuario por dispositivo de forma concurrente.

RNF (10).- El tiempo de respuesta de la aplicación no deberá de exceder de treinta segundos.

Requisitos tecnológicos

RNF (11).- La aplicación usará diseño responsivo y será compatible con dispositivos móviles y tabletas con conexión a Internet.

RNF (12).- Se ofrece la posibilidad de vincularse a redes sociales como Facebook y Twitter.

4. Diseño de la aplicación

Una vez conocida la funcionalidad que hay que implementar, es necesario explicar cómo se va a implementar. Primero se especificará la arquitectura elegida que resuelva mejor el problema, junto con sus herramientas utilizadas y posteriormente se presentarán los diagramas de diseño de la base de datos y los modelos de la aplicación.

4.1. Arquitectura web

Para explicar la arquitectura web de Codeps, es necesario elegir previamente las herramientas o tecnologías que se van a usar para desarrollar la aplicación, según los requisitos necesarios de la sección anterior. Al haber múltiples alternativas, es una fase importante ya que una buena o mala elección, repercute en las siguientes fases del proyecto.

4.1.1. Herramientas de Back-End

El sistema operativo que se va a utilizar en el lado del servidor Web es Windows 8. A simple vista puede ser un reto configurar varias herramientas de Back-End al mismo tiempo y que todo ello funcione correctamente y quizá pueda ser más rápido usar Linux (software libre), pero con el soporte y la compatibilidad que ofrecen los productos sobre Windows, es suficiente para instalar y configurar todo lo necesario.

Figura 8. Herramientas de Back-End

Para el desarrollo del proyecto, resulta indispensable la utilización de un servidor Web que atienda las peticiones de los clientes y responda con los contenidos correspondientes. IIS es uno de los servidores que mayor servicio ofrecen, pese a que actualmente ha sido superado por otros como NGINX, con buenas características en el apartado de balanceo de carga y tolerancia a fallos. Pero el servidor que se va a emplear es **Apache** por los siguientes motivos. Es un servidor Web de código abierto que implementa el protocolo HTTP/1.1 y uno de los más populares. Es multiplataforma y su arquitectura es muy modular y extensible, lo que permite de una manera muy sencilla ampliar sus capacidades.

Un caso concreto del servidor web Apache es el módulo **ModRewrite**, que se encarga de traducir, redirigir y modificar direcciones URL para hacerlas más amigables. Hoy en día, este módulo es una herramienta indispensable sobre todo de cara al posicionamiento en Google, ya que una URL es sensible a mayúsculas y minúsculas y no es un caso favorable al duplicar contenido. Aunque generalmente se utiliza para conseguir que las direcciones visibles sean sencillas de recordar y evitar las incómodas variables.

A la hora de elegir un lenguaje interpretado, existe una gran variedad: ASP.net, Java, Ruby, Python, PHP o incluso JavaScript también del lado del servidor (que puede resultar atractivo, al usar un mismo lenguaje en toda la aplicación). Pero se ha decidido usar **PHP Hypertext Preprocessor**, al ser el lenguaje más conocido (cursado en la propia universidad) y con mayor comunidad de usuarios detrás, que cuenta con una extensa biblioteca de funciones. PHP sirve para crear aplicaciones web dinámicas con acceso a información almacenada en una base de datos y que permite incorporar código en las páginas HTML de forma sencilla. Usa técnicas de programación orientada a objetos y sigue el patrón de diseño Modelo Vista Controlador. Es usado en millones de aplicaciones, entre las que destaca Wikipedia o Facebook. Además es un lenguaje soportado perfectamente por el servidor web Apache.

Para entenderlo mejor, PHP se comporta como un módulo de Apache, que extrae código dentro de las páginas, lo ejecuta en el servidor y envía el resultado al cliente. Éste no puede visualizar el código del programa, solamente su resultado. La Figura 9 muestra un ejemplo sencillo de su funcionamiento.

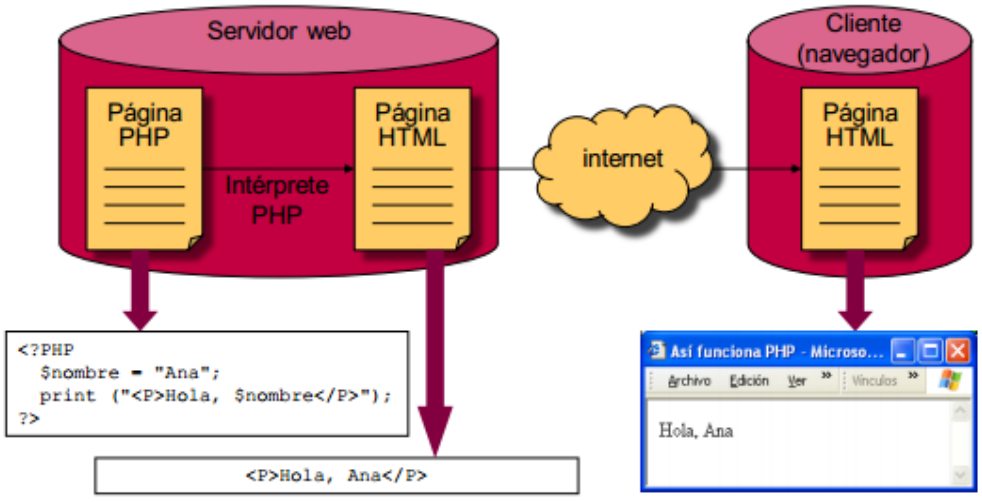

Figura 9. Funcionamiento de PHP

El servidor Web y en especial, este módulo PHP, puede conectarse a una base de datos para almacenar o extraer información a la hora de elaborar las páginas. Aquí también existen varias alternativas: SQL Server, MySQL, PostgreSQL o MongoDB, una base de datos NoSQL en la que se puede ejecutar JavaScript para realizar consultas. Aunque SQL Server sea una muy buena opción para Windows y MySQL sea una de las más utilizadas junto a aplicaciones con PHP a lo largo de estos últimos años, se va a utilizar **PostgreSQL**, un potente sistema de gestor de bases de datos relacional orientado a objetos y libre, del que se posee mayor conocimiento al ser utilizado en la universidad. Se ha tenido en cuenta que es uno de los servidores de bases de datos más lentos pero la aplicación Codeps no prevé tanto rendimiento. Entre sus ventajas se encuentran: funciones en distintos lenguajes, alta concurrencia sin bloqueos al acceder a los datos, buenos manuales y documentación, con una comunidad de desarrolladores activa bastante grande.

4.1.2. Herramientas de Front-End

Los contenidos devueltos por el servidor son interpretados por los navegadores Web de los clientes, que los muestran por pantalla acorde a unas fuentes y formatos. Es decir, el servidor Web se limita a transferir el código de la página sin llevar a cabo ninguna interpretación de ella. Aquí es donde intervienen las herramientas del lado del cliente. Una de ellas se trata del lenguaje HTML, un lenguaje de marcado que se ideó con el propósito de definir la estructura de una página o documento web, que tiene como finalidad reunir una serie de información en diferentes formatos (texto, imágenes, video, audio, etc.).

Figura 10. Herramientas de Front-End

Aunque la funcionalidad y estructura de una página web es sencilla, se puede complicar en exceso si se incluye en una misma página los contenidos, el diseño y la programación. La idea es separar la estructura de un documento de su presentación y diseño. **CSS** es un lenguaje utilizado para definir la presentación de un documento estructurado escrito en HTML.

Con el paso del tiempo, este lenguaje de hojas de estilo ha evolucionado mucho, permitiendo desde las tareas más elementales, como cambiar las dimensiones o los colores de un elemento, hasta efectos interactivos, transiciones o animaciones. Para aprovechar estas características sin requerir de un gran coste de tiempo adicional, se usa **Bootstrap 3.3.4**, un *framework* para CSS que ayuda a integrar al proyecto componentes prediseñados y adaptado para un diseño responsivo de la web. Es muy utilizado y es soportado por la mayoría de los navegadores web actuales, con algunas excepciones para versiones inferiores de Internet Explorer 9 (Tabla 3).

	Chrome	Firefox	Internet Explorer	Opera	Safari
Android	✔ Supported	✔ Supported	N/A	✖ Not Supported	N/A
iOS	✔ Supported	N/A		✖ Not Supported	✔ Supported
Mac OS X	✔ Supported	✔ Supported		✔ Supported	✔ Supported
Windows	✔ Supported	✔ Supported	✔ Supported	✔ Supported	✖ Not Supported

Tabla 3. Navegadores que soportan Bootstrap 3.3.4

La aparición de otras tecnologías, como el lenguaje de programación interpretado JavaScript, provocó que las páginas HTML también incluyeran el código de las aplicaciones que se utilizan para crear páginas web dinámicas, por lo que es conveniente la separación. Se va a usar el *framework* **jQuery 1.11.3** de JavaScript para ayudar a incorporar efectos dinámicos, animaciones, modificaciones del comportamiento en determinados eventos o ventanas con mensajes de aviso al usuario. Como se muestra en la Tabla 4, se usa esta versión porque es soportado por la gran mayoría de los navegadores.

	Internet Explorer	Chrome	Firefox	Safari	Opera	iOS	Android
jQuery 1.x	6+	(Current - 1) or Current	(Current - 1) or Current	5.1+	12.1x, (Current - 1) or Current	6.1+	2.3, 4.0+
jQuery 2.x	9+						

Tabla 4. Navegadores que soportan jQuery 1.11.3

4.1.3. El framework Laravel

Ponerse a desarrollar desde cero con estas herramientas puede parecer una tarea bastante complicada. Es por ello que se hace uso de *frameworks* generales que integren otras herramientas, como las especificadas anteriormente, para facilitar el trabajo e indicar al programador una estructura organizada de la aplicación para que luego sea posible un mejor mantenimiento. Se puede empezar mezclando código PHP en la vista y que el controlador termine realizando todo el trabajo o seguir un patrón que resuelva estos problemas desde el punto de vista de la ingeniería del software.

Figura 11. Uso de frameworks

Laravel es un potente *framework* de última generación, creado por **Taylor Otwel** en el año 2011, que facilita a los desarrolladores la creación de aplicaciones web de forma rápida con un código limpio, elegante, ordenado y con patrones profesionales para los usuarios más avanzados. Es decir, permite crear tanto aplicaciones sencillas como más complejas, de forma fácil y divertida. Pero, ¿por qué escoger Laravel y no otros *frameworks* PHP?

Laravel tiene una sintaxis mucho más sencilla de escribir y de entender, manteniendo el enfoque de PHP. *Frameworks* como Symfony o Zend son más aburridos al intentar parecerse más a Java, CakePHP a Ruby y CodeIgniter, como opinión personal, ha sido de los pocos que se ha mantenido en los primeros puestos de referencia. Tanto Laravel como CodeIgniter son fáciles de usar, pero éste último se quedó un poco anticuado. Las diversas formas de usar PHP por millones de usuarios, ha acabado perjudicando al lenguaje y haciéndolo menos seguro y profesional. Laravel surge para rescatarlo y volver a confiar en PHP, aprovechando al máximo las características de las últimas versiones, como el uso del **paradigma POO** (programación orientada a objetos). Laravel se adapta a proyectos sencillos y complejos, de forma más ordenada y estructurada, haciendo que las aplicaciones sean más fáciles de mantener. Destaca su profesionalidad y velocidad de desarrollo frente al resto de opciones: routes, closures, helpers, validación de formularios, middlewares, internacionalización, objetos ORM, motor de plantillas, etc. Todo esto se verá más adelante en el capítulo cinco.

Frameworks como Symfony siguen una arquitectura web orientada más bien a cliente-servidor. Sin embargo, la mayoría usa el **patrón MVC** (Model View Controller) que funciona bien en aplicaciones web y aquí es donde aparece Laravel para renovar el estilo de programación de PHP, usando una variante de MVC (Figura 12).

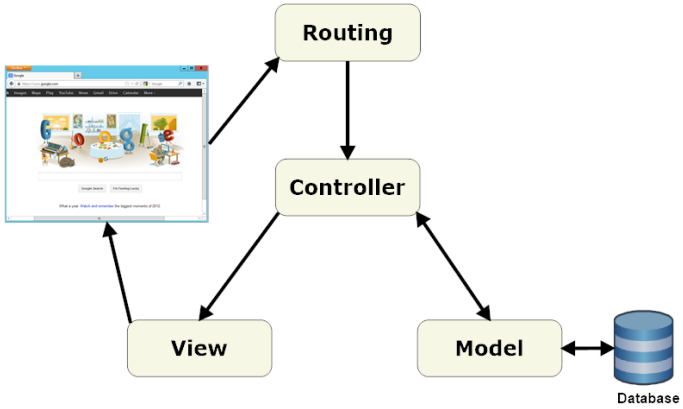

Figura 12. Patrón de diseño MVC con Laravel

En primer lugar, el navegador envía una petición al servidor. Laravel dispone de un sistema de enrutamiento que elige el controlador adecuado para procesar la solicitud. El controlador interactúa con el modelo para conectarse a la base de datos y recuperar o almacenar la información necesaria. Luego, el controlador envía los resultados a la vista. Por último, es la vista la que se encarga de entregar los resultados al navegador. El modelo de secuencia de la aplicación sería el que se muestra en la Figura 13.

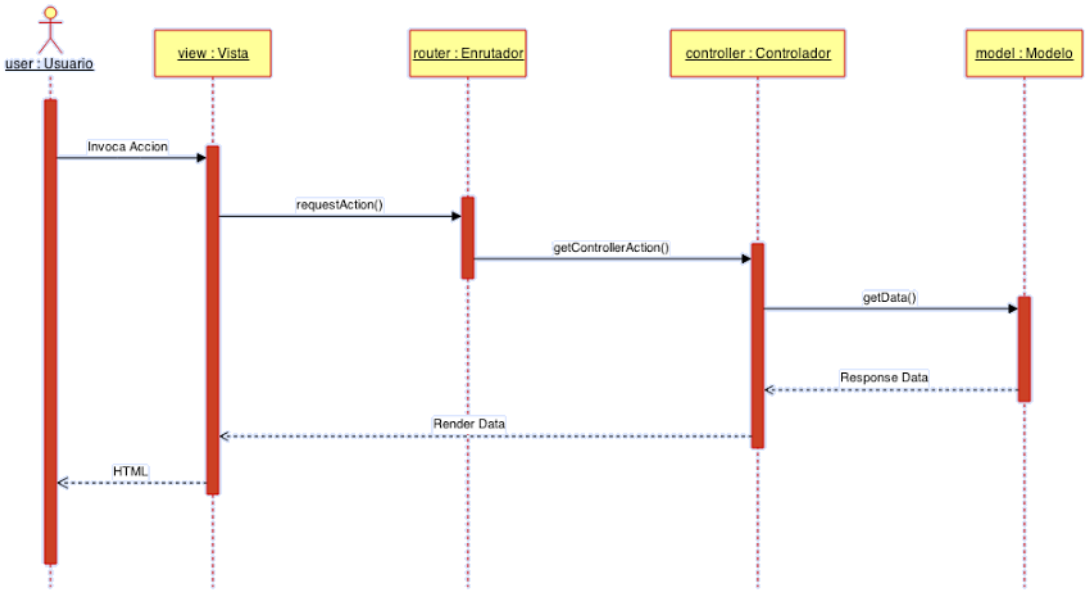

Figura 13. Diagrama de secuencia con Laravel

4.2. Estructura del proyecto

Es conveniente hacer un resumen antes de comenzar con el desarrollo de Codeps. El proyecto utiliza Apache, PHP, PostgreSQL y Laravel, integrado con Bootstrap y jQuery, además del IDE de desarrollo PhpStorm, organizando todo ello en los directorios de la Tabla 5.

Directorio	***Descripción***
D:\Servidor\apache	Servidor Web
D:\Servidor\pgsql	Servidor Database
D:\Servidor\php	Lenguaje de Back-End
D:\Servidor\phpstorm	IDE de PHP
D:\Servidor\tfg	Documentación
D:\Servidor\www	Sitio Web: phpPgAdmin, codeps (Laravel)

Tabla 5. Organización del proyecto

Se puede seguir la instalación y configuración de estas herramientas en los anexos del final de este documento. La aplicación Codeps se creará dentro de la carpeta *www* que es la ruta que el **servidor web Apache** tiene configurada para poder tener acceso a los datos del sitio Web. Los clientes podrán acceder al sitio a través del **navegador web Firefox**, por ejemplo.

Lo primero a tener en cuenta cuando se crea una aplicación con Laravel es la funcionalidad que tiene cada uno de los archivos y carpetas en los que está estructurado. Los sitios más utilizados son dos: *app* y *resources* (Tabla 6). En ***app*** es donde trabaja el desarrollador de back-end y se almacena el control de acceso a rutas, los controladores con lógica de negocio, los modelos encargados de persistir los datos en la base de datos, los middlewares para filtrar cualquier petición http con seguridad (si es requerido iniciar sesión o ser administrador).

En ***resources*** es donde trabaja el desarrollador de front-end: vistas html, motor de plantillas para interactuar con PHP y cargar información necesaria de la aplicación o de la base de datos, variables para cambiar el idioma o almacenamiento de recursos que no sean públicos.

Carpeta	*Descripción*
app	Back-End: routes, controllers, models, middlewares, requests, providers, services
bootstrap	Archivos necesarios para inicializar el framework
config	Archivos de configuración
database	Migrations y seeders
public	Hojas de estilo, javascript, imágenes, index.php
resources	Front-End: vistas, plantillas, idioma
storage	Archivos temporales, caché, logs de errores
tests	Pruebas con PHPUnit o PHPSpec
vendor	Componentes instalados de terceros

Tabla 6. Estructura de Codeps

En el directorio raíz se puede encontrar el archivo *composer.json*, con la información básica del proyecto y un archivo oculto donde se inicializan variables globales como el usuario y la contraseña de la base de datos. Rápidamente surgen varias cuestiones. ¿Cómo es posible ordenar todos los archivos del lado del servidor en una misma carpeta manteniendo el *framework* sencillo? ¿Cuál es el punto de inicio y su funcionamiento?

Aunque se puede ordenar libremente, la idea es separar la lógica del modelo de datos. En la carpeta `Http` estará el archivo de rutas y todos los controladores. De hecho, su creador recomienda eliminar la clasificación en modelos y crear capas con una funcionalidad en concreto. Con esta idea, cada clase de una capa (ya sea un repositorio, un manejador, un filtro, un evento, una colección o, en general, una utilidad) debe tener una única responsabilidad. Esto favorece el seguimiento del **patrón DRY** (Don't repeat yourself) con soluciones a problemas comunes como es el de repetir código que se va a utilizar en más ocasiones.

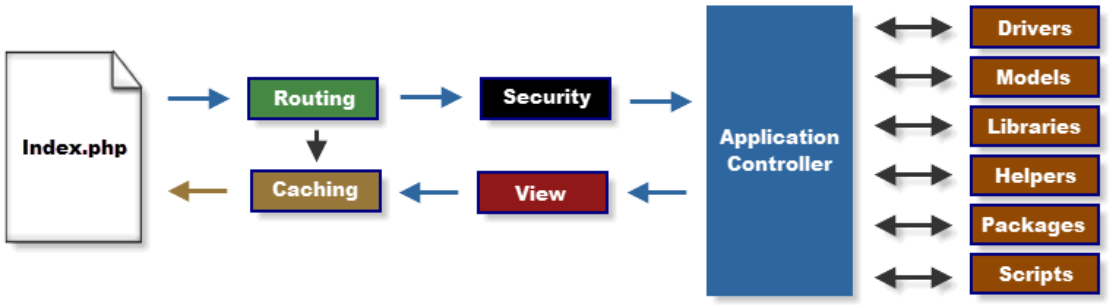

Figura 14. Patrón de diseño Front Controller

Cualquier petición del usuario se resuelve a través de las rutas definidas en `routes.php`. En Laravel existe un único punto de acceso a la aplicación Web, encargado de manejar estas peticiones, incluyendo a los servicios de seguridad como la autentificación y autorización, con el objetivo de elegir una vista apropiada o devolver un error como resultado. El **patrón Front Controller** (Figura 14) sugiere la centralización del manejo de peticiones en un controlador, pero a diferencia de Singleton, no se limita el uso de varios controladores.

Laravel 5 usa el **estándar PSR-4** para cargar todas las clases automáticamente. Para ello, es necesario que todas las clases de la aplicación usen al menos un espacio de nombre principal [8]. Con esto, no es necesario usar *composer dump-autoload* cada vez que se cree una clase.

El **patrón Decorator** responde a la necesidad de agregar funcionalidad a un objeto por medio de la asociación de clases. En Laravel, se puede usar distintas combinaciones de decoraciones para generar distintas páginas web y evitar repetir código cuando se define una vista en las plantillas. De esta forma se construyen formatos específicos como la cabecera (header), el pie de página (footer), un menú, una tabla de usuarios, etc., y se pueden añadir a una vista que, por ejemplo, represente la página principal.

Laravel usa un motor de plantillas llamado **Blade**, un lenguaje muy sencillo, que antes de ser usado por la aplicación, es compilado a PHP plano. A la hora de crear las plantillas en la carpeta `resources/views`, simplemente hay que añadir la extensión `.blade.php`. Dentro de una plantilla se puede extender de otra vista que tenga una sección dinámica sin completar (`@yield`) y escribir el código que se va a sustituir (`@section`). Las etiquetas de Blade `{{$n}}` permiten escapar los datos automáticamente, como seguridad, para protegerse de los ataques XSS (Cross-site scripting) [9]. Si no se desea escapar los datos, se puede usar `{!! $var !!}`. En `storage/framework/views` se guardan los archivos temporales con el HTML final.

La inyección de dependencias de Laravel se usa en los métodos constructores (`__construct`), en los métodos de los controladores y en las rutas, de forma automática, por lo que no hay que preocuparse en pasar las clases al método. Se usa el **patrón IoC** (Inversión de Control) donde un contenedor se encarga de resolver todas las dependencias que usa el objeto a crear, de forma transparente al usuario y se devuelve una instancia de dicho objeto [10].

Para finalizar esta sección, se muestra la navegación entre las diferentes rutas de Codeps. Desde la página principal se puede tener acceso a todas las vistas, pero su uso está restringido (filtros de autenticación en el sistema y de administrador). Para una mejor organización, la aplicación se divide básicamente en dos grandes módulos: usuarios y cursos. Cada uno de ellos cuenta con las mismas funcionalidades: una parte de administración (para editar usuarios y cursos almacenados) y otra parte de consulta (para interactuar con el perfil de usuario y con los cursos). Un usuario visitante no puede acceder a ninguno de estos módulos (Figura 15).

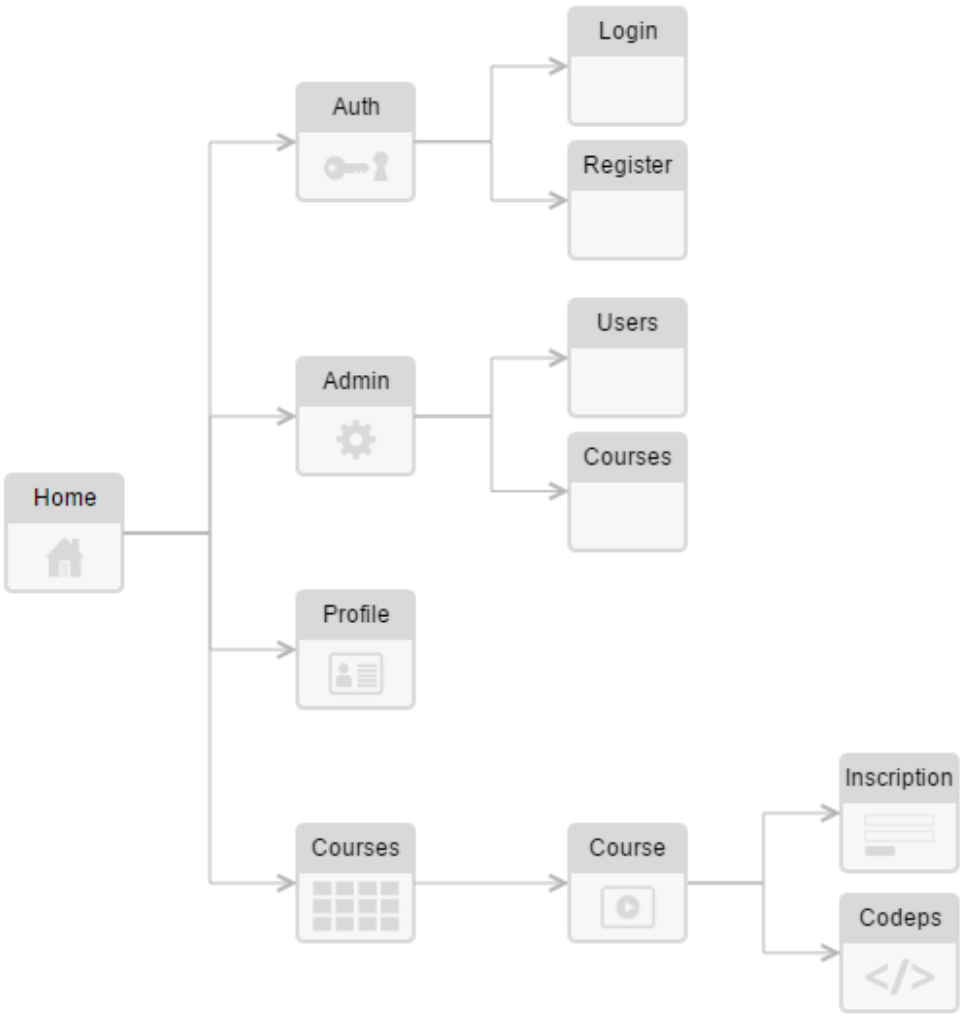

Figura 15. Diagrama de navegación de Codeps

4.3. Diseño de la base de datos

La aplicación Codeps usa una base de datos relacional para almacenar la información de todos los usuarios registrados en el sistema y todos los cursos que hay disponibles (Figura 16). Cada usuario tendrá su perfil, donde se indica si es un administrador o un estudiante. Cada curso será una versión de un tipo de curso específico, que tiene asignado una categoría del curso y contiene varios grupos a los que los alumnos pueden registrarse. Una versión del curso está formado por varios temas, éstos por varios capítulos y a su vez por distintos contenidos. Un contenido representa la unidad básica que puede aprender un estudiante en el curso y suele contener una explicación teórica junto con el ejercicio a resolver.

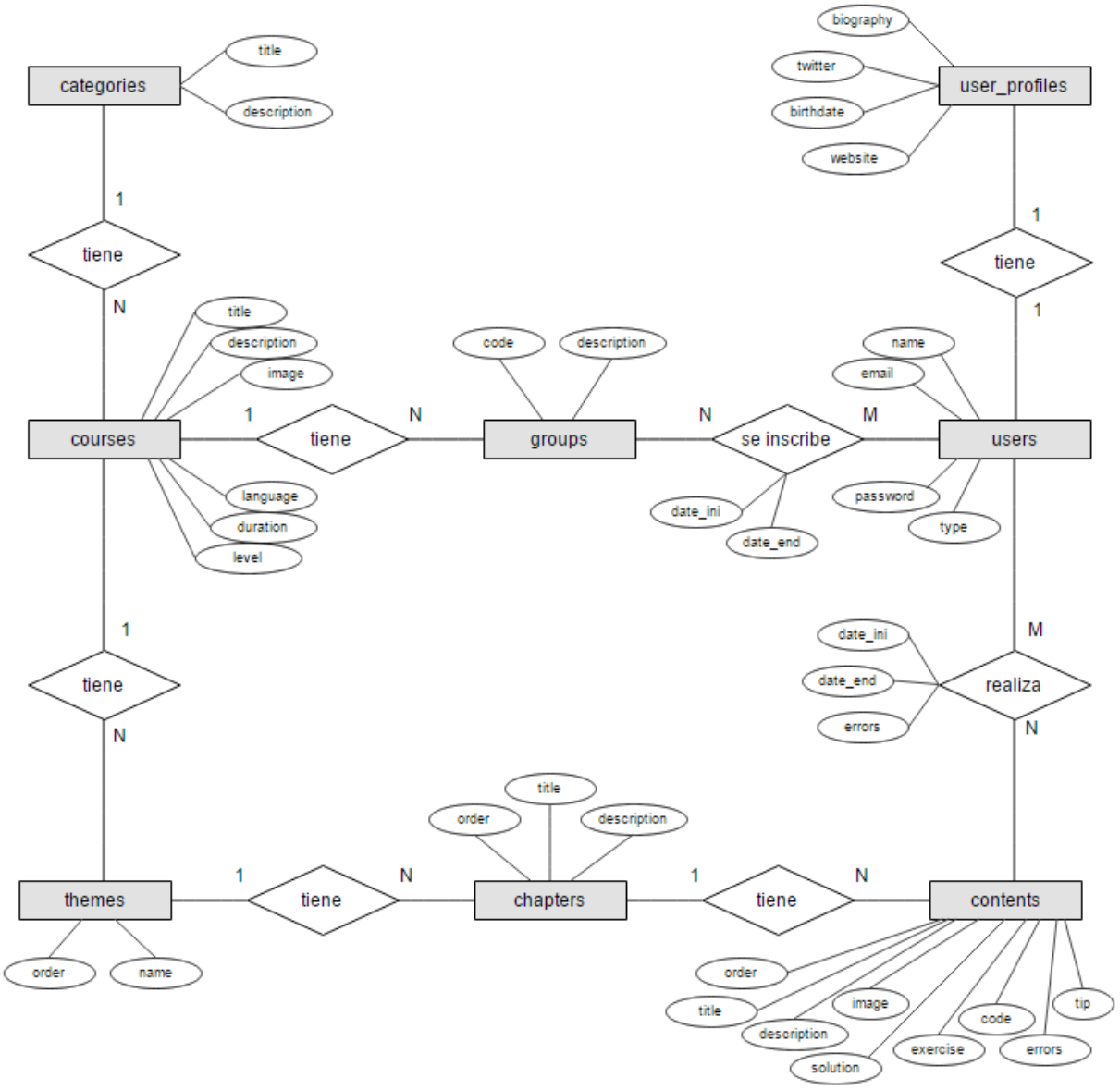

Figura 16. Modelo Entidad-Relación de Codeps

Los atributos de las entidades pueden variar con respecto a la versión final de la aplicación. Para conocer más acerca de los modelos de Entidad-Relación (tipos de entidades y relaciones disponibles) y el diseño de bases de datos relacionales por mapeado ER, así como la normativa a seguir según el tipo de relación, se pueden consultar los capítulos 3 y 7 de [11].

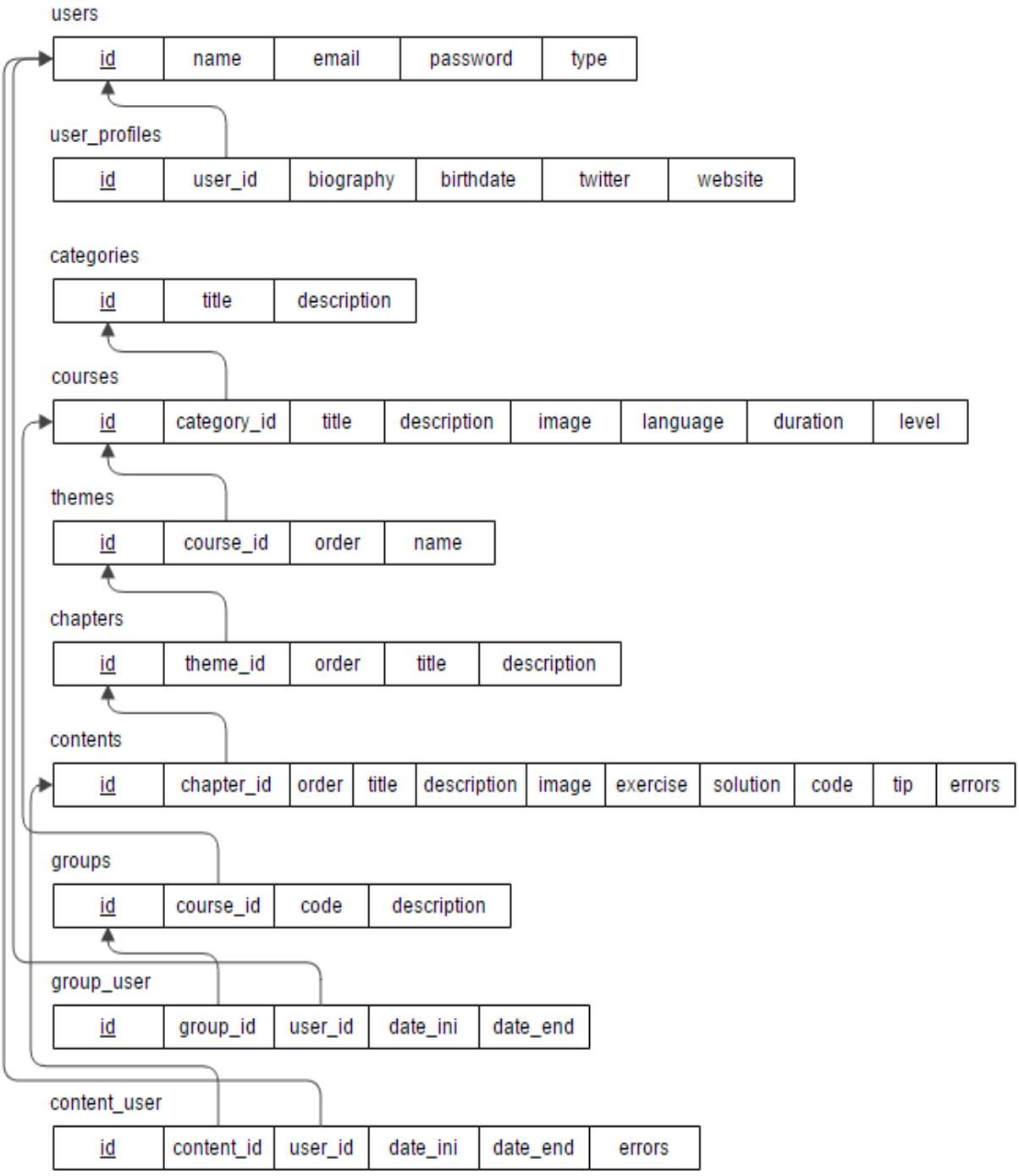

Figura 17. Esquema relacional de Codeps

Laravel usa el **patrón Active Record** para la persistencia de los datos. Se trata de una clase que se encarga de implementar todas las operaciones de consulta y modificación de una tabla de la base de datos, que a diferencia de otros patrones como DAO (Data Access Object), va a aportar menor flexibilidad a cambio de un mayor aislamiento para trabajar con el sistema de gestión de la base de datos, en este caso PostgreSQL.

El patrón Active Record permite trabajar las tablas como si fueran clases y las filas como objetos. Para ello, Laravel cuenta con un potente ORM (Object-Relational mapping) llamado Eloquent y además tiene un constructor de consultas SQL (query builder) llamado Fluent. También permite utilizar SQL puro para consultas más complicadas.

En primer lugar, **Fluent** es un constructor de consultas SQL, basado en PDO, encargado de generar cualquier consulta a la base de datos. Las consultas generadas vienen por defecto con los niveles de seguridad para evitar inyecciones SQL de usuarios malintencionados [12]. Las bases de datos relacionales son incompatibles con la forma de trabajar en la programación orientada a objetos, en la que los objetos se relacionan con otros a través de propiedades y métodos. **Eloquent** permite mapear datos de la base de datos y convertirlos a objetos o, por el contrario, tomar un objeto y almacenarlo como un registro de la base de datos.

En el siguiente capítulo se explica la forma de usar estas herramientas y además se introduce el concepto de *migraciones*, para llevar un control de versiones de la base de datos, y *seeders*, para cargar información aleatoria en las tablas y poder probar la aplicación.

4.4. Diseño web adaptable

Tener un sitio especializado para cada uno de los dispositivos existentes puede llegar a ser muy complicado de mantener, ya que no se cubre la variedad de pantallas y dispositivos móviles disponibles hoy en día. Google publicó recientemente que su algoritmo de búsqueda incluirá el factor de diseño RWD (Responsive Web Design) viéndose afectados todos los sitios que no sean adaptables.

Para lograr un diseño adaptable existen diferentes técnicas relacionadas con HTML, CSS y JavaScript. El framework Bootstrap dispone de **CSS media queries** para escalar de manera fácil y eficiente los sitios web y aplicaciones, usando una base de código sencilla, válido desde teléfonos y tabletas a dispositivos de escritorio.

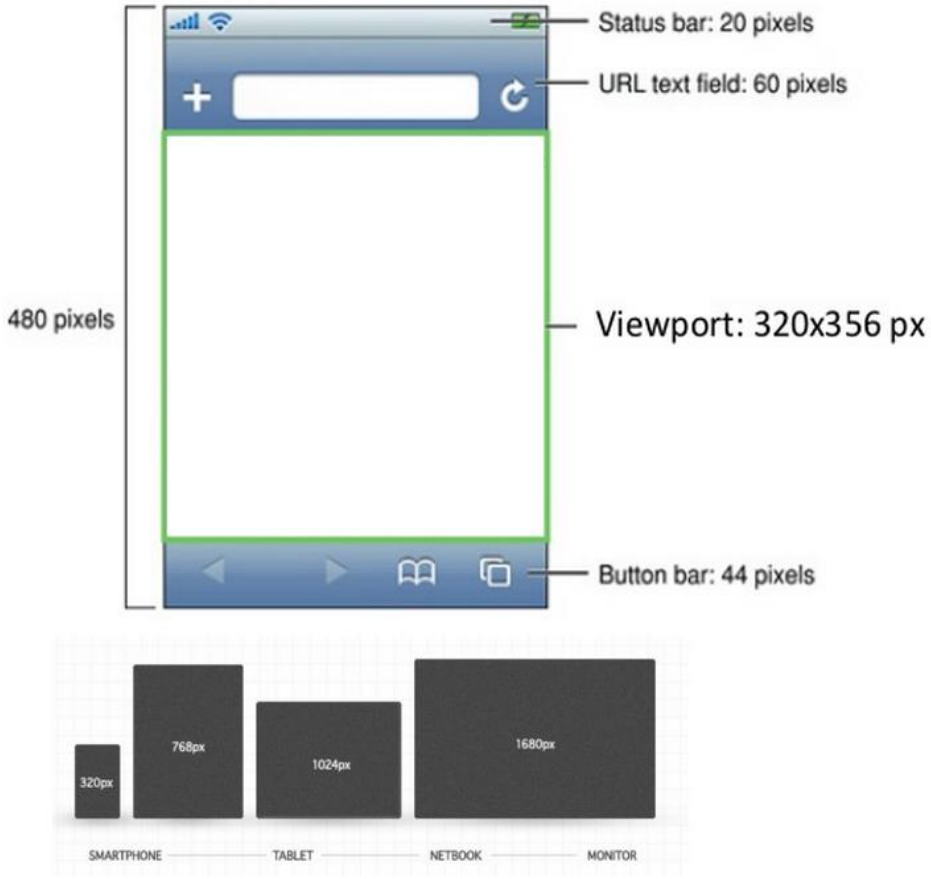

Figura 18. Diseño web adaptable

Incluyendo la regla: `<meta name="viewport" content="width=device-width, initial-scale=1">` se adapta el contenido de la página al ancho del dispositivo, permitiendo el zoom. Es importante destacar que los píxeles que usa la etiqueta *viewport* son píxeles CSS, que pueden variar con respecto a los píxeles físicos del dispositivo. Esto es debido a la diferencia en la densidad de píxeles que tenga la pantalla. Existen herramientas para calcular la dimensión exacta en píxeles CSS [13]. El framework Bootstrap usa un sistema de rejillas para organizar todos los contenidos de la página, donde cada fila contiene doce columnas. Las columnas (`div class="col-md-4"`) se declaran dentro de una fila (`div class="row"`) y cada fila tiene que estar dentro de un contenedor (`div class="container"`).

Los *media queries* son un método para detectar ciertas características del *viewport*, como son sus dimensiones (width y height), la resolución y la orientación del dispositivo (portrait o landscape). Bootstrap aplica este método en sus hojas de estilo usando la regla `@media` seguido de las condiciones que debe cumplir el *viewport*, permitiendo mayor flexibilidad.

5. Implementación

Antes de empezar a desarrollar la aplicación, es necesario explicar algunos conceptos que incluye Laravel. Los principales elementos básicos de Laravel son Composer, Artisan, routing, controllers, requests, responses y views. En relación con la base de datos hemos de explicar migrations, seeders, Fluent y Eloquent. Finalmente, en relación con la arquitectura repasaremos service providers, containers, contracts y facades. Posteriormente, se explican varios de los servicios que incluye el framework (autenticación, cache, colecciones, encriptación, eventos, internacionalización, mail, paginación, sesión, validación, motor de plantillas Blade, etc.), además de aquellos creados especialmente para cada uno de los módulos en los que se divide Codeps.

5.1. Primeros pasos con Laravel

En los anexos de final del documento, se indican los manuales de instalación del entorno de desarrollo: Apache, PHP, PostgreSQL y Laravel, necesarios para el desarrollo de la aplicación Codeps. Durante la instalación del framework es necesario usar la herramienta **Composer**, un manejador de dependencias de PHP. Actualmente es una de las mejores herramientas que hay disponibles para el desarrollo con el lenguaje PHP. Permite manejar de una forma sencilla las dependencias del proyecto, ya sean del framework completo o de cualquier otro componente que instalemos. La herramienta se encarga de instalar todas las dependencias de manera recursiva, manteniendo los paquetes actualizados y de forma automática.

Figura 19. Manejador de dependencias Composer

Laravel usa el estándar PSR-4 que permite a Composer cargar las clases automáticamente, ya que lo combina con los *namespaces* de PHP para ordenar y estructurar mejor el proyecto, por lo que, una vez creado el proyecto de Laravel con Composer, lo segundo que habrá que hacer es configurar `composer.json` para que los componentes usen esta especificación, donde la clave es el espacio de nombres y el valor es el nombre del directorio.

```
"autoload": {
      "psr-4": {
            "App\\": "app/"
      }
}
```

Artisan es el nombre de la interfaz de línea de comandos incluida en Laravel. Para mostrar una lista de todos los comandos que hay disponibles: `php artisan list`, y para obtener ayuda: `php artisan help comando`. Por ejemplo, para cambiar el nombre de la aplicación: `php artisan app:name nombre`. Los nombres de espacio se deben asignar a todas las clases para poder diferenciarlas en el proyecto. Una de las ventajas de usar PSR-4 es que no requiere ejecutar: `composer dump-autoload` cada vez que se crean clases dentro del directorio `app`.

En la carpeta `config` están los archivos de configuración que contienen las instrucciones necesarias para cargar y configurar el framework. El archivo `config/app.php` es el lugar central de la aplicación, donde Laravel especifica los **Services Providers** que hay que cargar y en qué orden aplicarlos. Los Services Providers son clases que se encargan de configurar otras clases y la manera en la que interaccionan entre sí con inyección de dependencias. Los que se crean para la aplicación están en `app/Providers`, mientras los que son del framework, o realizados por terceros, en `vendor`. El archivo `config/database.php` contiene las variables para configurar la base de datos a la que se va a conectar. Para máxima seguridad, en el archivo oculto `.env` se declaran los datos, como las contraseñas, que serán privados y se accede a ellos mediante el *helper*: `env('clave', 'valor_por_defecto')`.

Todas las peticiones del cliente llegan al archivo `public/index.php` y luego son interpretados por el archivo de rutas `app/Http/routes.php` donde se definen los **Controllers** que se encargan de procesar la petición (para casos sencillos, se pueden usar **Closures**, funciones anónimas donde se puede llamar directamente al *helper*: `view('vista')`. Laravel cuenta con HTTP Requests y HTTP Responses para resolver las solicitudes, que se envían por un formulario o al subir un fichero, de forma sencilla.

A continuación, se instala el sistema de migraciones: `php artisan migrate:install` para llevar un control de versiones de la base de datos, tanto para crear las tablas como para realizar cambios durante la implementación. Las migraciones permiten, no sólo definir las tablas con programación orientada a objetos en vez de SQL, sino tener una gran portabilidad a distintos servidores de base de datos y a cambios que se pueden producir posteriormente. Para cargar datos de prueba rápidamente en las tablas se usan *seeders*: `php artisan db:seed`.

Nota: la información no se guarda al hacer una migración; cada vez que se hace un cambio en las migraciones o *seeders* hay que ejecutar `composer dump auto-load`, ya que éstos usan *classmap* en vez del patrón PSR-4.

Antes de comenzar a implementar cada uno de los módulos, hay que realizar una última configuración: la **internacionalización** de la aplicación. Laravel permite de una forma sencilla cambiar el idioma de los textos literales modificando una variable de localización dentro de `config/app.php`. Para cambiar su valor dinámicamente se usa: `App::setLocale('es')`. En la carpeta `resources/lang` se guardará los archivos de los idiomas que contiene todas las traducciones posibles. Para referirse a una variable desde el código, hay que usar siempre el siguiente *helper*: `trans('archivo.clave');` (es un alias del método `Lang::get('archivo.valor'))`, o desde Blade: `@lang('archivo.clave')`. Para más información se puede consultar en [14] y [15].

Tras esta breve introducción, se crean las primeras páginas de la aplicación. Lo primero será crear una página maestra (`master.blade.php`) que contenga la cabecera, el pie de página y los scripsts para integrar los frameworks Bootstrap y jQuery. Laravel incluye por defecto la primera página (`index.php`) que se encarga del enrutamiento. A partir de ahora, todas las páginas extenderán de la página maestra y sólo hará falta definir el contenido, que es la parte dinámica a mostrar. En este primer apartado, se crea la página principal (`home.blade.php`), que es una vista sencilla de bienvenida, la página de soporte (`contact.blade.php`) que contiene un formulario de contacto donde los usuarios envían un correo a la administración del sistema y la página de ayuda (`about.blade.php`) que muestra información de interés.

Se usan dos *helpers* bastante útiles en las plantillas de Blade: `url()` y `asset()` para conseguir los enlaces absolutos de las direcciones web y de los archivos de la carpeta pública. También es necesario instalar el componente de **Laravel Collective** [16] para utilizar las etiquetas dinámicas de HTML y FORM, como por ejemplo `Html:style()` o `Form::text()`. Su instalación se puede aplicar a cualquier otro componente disponible para Laravel. En primer lugar, se añade en composer.json:

```
"require":
{
    "laravelcollective/html": "5.1.*"
}
```

Desde la terminal: `composer update`. Se añaden en `config/app.php` los *providers* y *aliases*:

```
Collective\Html\HtmlServiceProvider::class,
'Form' => 'Collective\Html\FormFacade',
'Html' => 'Collective\Html\HtmlFacade',
```

Para cargar la página principal, se crea un controlador que se encargue de su comportamiento.

```
php artisan make:controller HomeController
```

Se edita de tal forma que en la función `index`, la respuesta sea `return view('home')`, que es un *helper* de `\View::make(plantilla_blade)`; y luego se añade la ruta en el archivo: `app/Http/routes.php`

```
Route::get('/', ['as' => 'home', 'uses' => 'HomeController@index']);
```

De esta forma, se indica que cuando se solicite http://codeps.es se está accediendo con método GET cuyo alias de la ruta es 'home' y se ejecuta el método 'index' por defecto.

5.2. Módulo de usuarios

Antes de crear un sistema de autenticación seguro, es necesario crear las tablas en la base de datos. Esto se puede lograr fácilmente a través de migraciones, que lleva un control de versiones de la base de datos *Codeps*. Cuando se instala el sistema de migraciones de Laravel, desde consola: `php artisan migrate:install`, se crea una tabla *migrations* que verifica las migraciones que ya se han realizado y no hace falta volver a ejecutar.

Para crear una migración: `php artisan make:migration create_users_table --create="users"`. Este archivo contiene dos métodos, donde se puede hacer uso de **Schema Builder** [17] para crear la tabla `usuarios`. Cuando se trabaja en local y se hace algún cambio en la tabla (se añade un nuevo atributo por ejemplo), es recomendable usar `php artisan migrate:rollback` para eliminar la tabla por el método DOWN y crear una nueva con el método UP. Si por el contrario, ya se ha subido el proyecto a un repositorio o al servidor compartido, se puede modificar la tabla ya creada, usando las nuevas migraciones `php artisan migrate`. De una forma más rápida, se puede usar: `php artisan migrate:refresh –seed`, que resetea la base de datos, crea de nuevo las migraciones y ejecuta los *seeders* para rellenar los datos.

Nota: tanto *rollback* como *reset* elimina todos los datos de la base de datos, por lo que es recomendable hacer una copia de respaldo con `pg_dump` por ejemplo, para el caso de Postgre.

Si no existe un tipo de datos en Schema Builder al cambiar de un gestor de base de datos a otro, Laravel lo sustituye por el tipo de datos más parecido que exista, lo que supone una gran ventaja ya que no hace falta crear nuevas instrucciones de actualización, simplemente con ejecutar el sistema de migraciones, Laravel crea las tablas correspondientes.

Una vez creadas las tablas de usuario y perfil de usuario por medio de las migraciones, se crean los formularios login.blade.php y register.blade.php, necesarios para el sistema de **autenticación** de la aplicación. En el archivo de rutas se añade:

```
Route::controllers([ 'auth' => 'Auth\AuthController', ]);
```

De esta forma, no se están creando unas rutas específicas, sino que se agrupan todos los métodos del controlador indicado, por la ruta `'auth'`. Es decir, el controlador tiene el método `getLogin()` para cargar la vista del formulario a través de GET cuando se accede a la url: `/auth/login` y el método `postLogin()` en el caso de datos de envío del formulario a través del método POST, donde se procesan esos datos (Request) por ejemplo, validándolos y creando una cookie con los datos de la sesión, en caso de ser correctos. Tanto para el caso de inicio de sesión como para el registro, se crea una variable `protected $redirectTo = '/';` que sirve para redireccionar el control a la vista principal.

Cuando se crea un usuario en el sistema, el método del controlador usa **Eloquent** para almacenar los datos recibidos del formulario. Por ello, la siguiente tarea es crear los modelos de usuario, en este caso `User` y `UserProfile`. Se sigue la notación *CamelCase* para todos los modelos y *snake_case* para los nombres de las tablas (por ejemplo, si se llama a la tabla: `users` y al modelo: `User`, no hace falta indicar más datos al ORM). En el directorio `app` se irán guardando todos los modelos creados con `php artisan make:model User`.

Una vez visto el modo de autenticación, se crea el de **administración**, donde los usuarios que son administradores pueden realizar otras tareas adicionales, como modificar, crear o eliminar usuarios y cursos. En este caso, no se necesitan ni rutas específicas ni rutas indicadas por el controlador, sino que se usan rutas agrupadas en un nombre de espacio, con la funcionalidad **RESTful**, un conjunto de operaciones CRUD (crear, obtener, actualizar y borrar) que se aplican a todos los recursos de información para su persistencia la base de datos.

```
Route::group(['prefix' => 'admin', 'namespace' => 'Admin'],
             function () { Route::resource('users', 'UsersController'); }
);
```

El modelo de usuario ya está creado, por lo que el siguiente paso es crear las vistas asociadas a cada operación. Se recuerda que para ver todas las operaciones disponibles asociadas a los métodos que se crean en el controlador, se puede usar: `php artisan route:list`. La vista principal (`index.blade.php`) se muestra al acceder desde el menú de administración de usuarios. Se crean también las vistas de creación y actualización de un usuario.

Método	***Path***	***Nombre ruta***	***Acción***	***Middleware***
GET	admin/users	admin.users.index	UsersController@index	auth, admin
GET	admin/users/create	admin.users.create	UsersController@create	auth, admin
POST	admin/users	admin.users.store	UsersController@store	auth, admin
GET	admin/users/{id}	admin.users.show	UsersController@show	auth, admin
GET	admin/users/{id}/edit	admin.users.edit	UsersController@edit	auth, admin
PUT	admin/users/{id}	admin.users.update	UsersController@update	auth, admin
DELETE	admin/users/{id}	admin.users.destroy	UsersController@destroy	auth, admin

Tabla 7. RESTful Resource Admin/UsersController

Los navegadores sólo permiten peticiones GET o POST, por lo que cuando se quiere enviar un verbo PUT o DELETE más la URL, se crea un campo oculto llamado `_method` con el valor PUT o DELETE, mientras que el método por defecto en el formulario es PUT.

Para reusar una lógica de consulta en un modelo y no repetir código, se pueden añadir atributos dinámicos usando la sintaxis `setNombreAttribute()` o también se pueden usar los **Query Scopes** de Eloquent; `scopeNombre($query, $arg)`, donde se puede seguir filtrando en la consulta, a través del método `where()` según el argumento recibido, en caso necesario.

Laravel usa inyección de dependencias y automáticamente se cargan todas las dependencias y se crean los objetos necesarios para trabajar en todos los métodos de la clase. Además, aunque se pueden usar *facades*, *alias* o *helpers* para cargar un objeto puntualmente, Laravel también

permite usar inyección de dependencias en el método para conseguir el mismo propósito. Es recomendable usar dependencias en el controlador cuando se trabaja en una clase especial, donde todos sus métodos requieren de ese objeto, mientras que las otras opciones se dejan para momentos puntuales dentro de un controlador por ejemplo, donde se quiere acceder a la funcionalidad de un objeto de forma rápida y sencilla.

Se usan los **Form Request** como inyección de dependencias en el método, para validar los datos de un formulario: `php artisan make:request CreateUserRequest`. En casos más complejos con muchos atributos a validar, se puede realizar fuera del controlador. En el método `rules()` se indican las reglas que tiene que cumplir cada atributo para ser válido. Esta opción tiene como ventaja que Laravel, de forma mágica y sin agregar líneas de código, se ocupa automáticamente de hacer las validaciones indicadas en las reglas cuando detecta un objeto Request en los argumentos del método. Si no se cumple alguna de las reglas, se redirigirá a la vista que estaba anteriormente.

Cabe destacar que cuando se accede al panel de administración de usuarios, se desea que su acceso esté restringido. Eso se consigue de una manera muy sencilla usando **Middleware**, código que se va a ejecutar antes de procesar una acción en el controlador. Para crearlo, se puede usar `php artisan make:middleware IsAdmin` y luego se tiene que registrar en el archivo `app/Http/Kernel.php`. Se puede registrar de manera global (se ejecutará siempre cuando se cargue una ruta) o crear una ruta:

```
protected $routeMiddleware = [
     'auth'       => \App\Http\Middleware\Authenticate::class,
     'auth.basic'  => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
     'guest'      => \App\Http\Middleware\RedirectIfAuthenticated::class,
     'admin'      => \App\Http\Middleware\IsAdmin::class,
];
```

Este nombre se puede usar en diferentes sitios: en el constructor de un controlador, por ejemplo: `$this->middleware('guest', ['except' => 'getLogout']);` o en un grupo de rutas. Sólo se desea dejar acceso al panel de administración a usuarios que hayan iniciado sesión y que además sean de tipo administrador, luego este último caso es más eficiente:

```
Route::group([
    'prefix' => 'admin',
    'middleware' => ['auth', 'admin'],
    'namespace' => 'Admin'],
    function () Route::resource('users', 'UsersController', ['except' => 'show']);
} );
```

Dentro del middleware hay que completar el método `handle()` que procesa la acción a realizar, donde en este caso se recupera el usuario que hay en la sesión y se comprueba si su tipo es administrador. Gracias a los **Contracts**, es muy sencillo obtener los datos del usuario que está autenticado. Se hace uso del contrato `Auth\Guard` de la siguiente manera:

```
if (! Auth::user()->isAdmin()) {
      Auth::logout();
      return redirect()->to('auth/login');
}
return $next($request);
```

Si no es administrador se desconecta por seguridad y se redirige a la vista de inicio de sesión, en caso contrario, logra pasar el filtro y sigue con la siguiente petición.

El usuario, ya sea estudiante o administrador (por lo que sólo hace falta incluir el middleware `auth`) tendrá acceso a la página de perfil de usuario, donde podrá editar sus preferencias como su biografía, dirección, edad y otros datos no obligatorios. La tabla de perfiles es una relación 1 a 1 con la tabla de usuarios, es decir, un usuario puede tener un perfil y un perfil pertenece a un usuario. La relación fuerte está en la tabla perfil, por lo que hay que incluir un atributo user_id como clave foránea. En Laravel, se puede usar Fluent para obtener los datos usando *joins*, pero Eloquent usa una manera más sencilla (y casi mágica) de relacionar los modelos sin escribir apenas código. Para relaciones **One To One** hay que crear los métodos:

```
class User extends Model {
    public function profile() {
        return $this->hasOne('App\UserProfile');
    }
}
class UserProfile extends Model {
    public function user() {
        return $this->belongsTo('App\User');
    }
}
```

Para obtener los datos de un perfil, se recupera el usuario de la sesión y se llama al método `\Auth::user()->profile()->first()` que devuelve una colección y ya es parte de la vista hacer uso de ellos. También se puede usar dinámicamente: `\Auth::user()->profile`.

Tanto en el caso del registro, como a la hora de crear un usuario, es necesario crear también un perfil `$profile = new UserProfile();` y guardarlo desde el usuario:

```
$user->profile()->save($profile);
```

Otra opción es asociar el usuario desde el perfil:

```
$profile->user()->associate($user);
$profile->save();
```

5.3. Módulo de cursos

La implementación de los cursos sigue la misma idea que la de los usuarios. Existe un panel de administración de cursos donde el administrador podrá editar y añadir nuevos cursos, así como mostrar información de ellos a través de estadísticas, y un perfil donde los usuarios podrán ver sus cursos inscritos y el porcentaje completado.

Los pasos a seguir a la hora de crear un nuevo módulo son muy parecidos al anterior, por lo que, para no extenderse demasiado, se pasa a hablar directamente de las relaciones en Laravel, que es donde reside la mayor complejidad, al tratarse de relaciones uno a muchos, donde un curso puede tener varios temas, un tema puede tener varios capítulos y un capítulo consta de contenidos docentes. Pero antes de continuar, conviene hacer un resumen de lo visto hasta ahora sobre lo básico de este *framework*.

Primero hay que crear las migraciones para indicar los atributos de las tablas de la base de datos y poder llevar un control posterior sobre ellas. Lo siguiente será crear el espacio de rutas para enrutar una URL con su acción. Pueden ser funciones anónimas (Closures) o métodos de los Controllers. Hasta aquí, es necesario tener creados los modelos y los controladores. En los controladores es donde se incluye toda la lógica (Middleware, Requests, Responses, Service Providers, Contracts, Facades) y servicios (Fluent, Eloquent, autenticación, encriptación, sesiones, validaciones, paginación, caché, almacenamiento, envíos de correo, eventos, etc.) disponibles en Laravel. Por último, hay que definir una vista para mostrar los resultados que se reciben como parámetros desde el controlador. Se puede usar el motor de plantillas Blade. Toda la documentación detallada se puede encontrar en la página oficial de Laravel [17] y en la nueva API de referencia [18].

Con todos estos conceptos básicos más claros, se continúa con el módulo de cursos. Una de las tareas que puede tener dificultad, es la subida de archivos o imágenes al servidor. Para ello, se hace uso del servicio de almacenamiento *Storage* de Laravel. En el archivo de configuración `config/filesystems.php` se cambia el directorio raíz a la carpeta pública, para poder visualizar las imágenes mostradas desde la vista `'root' => public_path()`.

Cuando se almacena una imagen, se tiene que comprobar que existe el archivo:

```
$course = new Course($request->all());
if (isset($request->file('image'))
{
    $nombre = $file->getClientOriginalName();
    $course->setAttribute('image', $nombre);
    $contents = file_get_contents($file->getRealPath());
    \Storage::disk('local')->put('images/courses/'.$nombre, $contents);
}
$course->save();
```

Ahora, desde la vista donde se van a mostrar los cursos, se recorren de la siguiente manera:

```
@foreach($courses->where('category_id', $category->id) as $course)
    <a href="{{ route('courses.show', $course->id) }}" class="thumbnail">
        {!! Html::image('images/courses/'.$course->image, 'alt') !!}
        <div class="caption text-center">
            <h3>{{ $course->title }}</h3>
            <p>{{ $course->description }}</p>
        </div>
     </a>
@endforeach
```

Si se desea obtener una información más completa se puede consultar en [19] y [20]. En el archivo de rutas, se agregan las rutas de tipo *resource* en el grupo creado para los usuarios, ya que comparten el mismo prefijo de administrador.

```
Route::group([
 'prefix' => 'admin',
 'middleware' => ['auth', 'admin'],
 'namespace' => 'Admin'],
   function () {
     Route::resource('users', 'UsersController', ['except' => 'show']);
     Route::resource('courses', 'CoursesController');
     Route::resource('courses.themes', 'ThemesController', ['except' => 'show']);
     Route::resource('courses.themes.chapters', 'ChaptersController', ['except' => 'show']);
     Route::resource('courses.themes.chapters.contents', 'ContentsController');
});
```

A continuación, se explica las relaciones **One To Many** en Eloquent, que son necesarias para trabajar con los cursos. Por ejemplo, un curso tiene muchos temas y un tema pertenece a un curso. La relación que almacena los temas debe contener una clave foránea al usuario.

```
class Course extends Model {
    public function themes() {
        return $this->hasMany('App\Theme');
    }
}
```

```
class Theme extends Model {
    public function course() {
        return $this->belongsTo('App\Course');
    }
}
```

En cada modelo (grupos, cursos, temas, capítulos y contenidos) se debe añadir estos métodos para que Laravel pueda relacionarlo con el ORM. Así, de esta forma sencilla, se puede consultar los temas que tiene un curso: `$course->themes()->get()` o también se puede utilizar su propiedad dinámica: `$themes = $course->themes`. Incluso, como el resultado es una colección de datos, se pueden seguir añadiendo filtros a la consulta.

```
$theme = $course->themes()->where('id', 1)->first();
$themes = $course->themes()->orderBy('id', 'DESC')->get();
```

Por ejemplo, si se quiere mostrar un curso detallado, una vez que se obtiene el curso, el controlador devuelve la vista que organizará el contenido y en ella se puede usar el parámetro recibido para recorrer todos los temas que tiene el curso.

```
$course = Course::find($id);
return view('course.show', compact('course'));
```

```
@foreach($course->themes as $theme)
    <h2>{{ $theme->name }}</h2>
    @foreach($theme->chapters as $chapter)
        <h4> {{ $chapter->title }}</h4>
        [...]
    @endforeach
@endforeach
```

Nota: las consultas Eloquent se componen de tres partes: `Model::constraint()->fetch()`. Para consultas más complejas se puede usar el facade con los métodos select, update, insert, delete y statement: `\DB::select('select * from users where type = ?', ['admin'])`.

Modelo	***Restricciones***	***Métodos de obtención***
Obligatorio	Opcional	Obligatorio
Ejemplos: User, Theme, Course	Ejemplos: where, orWhere, whereRaw, select, orderBy, take …	Ejemplos: all, find, first, get, toSql, count, max, update, delete …

Tabla 8. Consultas en Eloquent

5.4. Inscripción de un usuario en un curso

En la aplicación, aunque 'inscribirse' y 'registrarse' pueden parecer sinónimos, se usará el término de registro cuando un alumno se registre en la aplicación como usuario, mientras que la palabra inscripción quedará reservada para referirse al registro de un alumno en un curso.

El acceso a una inscripción se puede realizar desde los detalles cuando se selecciona un curso, si un estudiante no está matriculado en ningún grupo, se muestra un botón para cargar una vista modal que permite elegir un grupo si está disponible. Al aceptar, la acción envía el código del grupo y es recibido por un controlador. Por tanto, la ruta creada sería:

```
Route::group([
  'middleware' => 'auth',
  'namespace' => 'Course'],
   function () {
     Route::resource('courses', 'CoursesController', ['only' => ['index','show','store']]);
});
```

El método `index` del controlador muestra todos los cursos disponibles, el método `show` muestra el detalle de cada curso y da opción a inscribirse, acción de la que se encarga el método `store`. Cuando se trabaja con relaciones muchos a muchos, es necesario crear una tabla intermedia llamada *pivote*, siguiendo la nomenclatura: nombre de las dos tablas de la relación en singular, en orden alfabético y separado por un subrayado (`group_user`). En Eloquent, no hace falta crear el modelo de la tabla pivote, lo que permite gestionar las relaciones **Many To Many** de manera sencilla. Se añaden los siguientes métodos:

```
class User extends Model {
  public function themes() {
    return $this->belongsToMany('App\Group')->withPivot('date_end')->withTimestamps();
}}
class Theme extends Model {
  public function course() {
    return $this->belongsToMany('App\User')->withPivot('date_end')->withTimestamps();
}}
```

Y ya se puede acceder a los atributos de la relación y a los de la tabla pivote: `$column = $group->users()->first()->pivot->end_date;`. En el método store, antes de asociar la relación, se recupera de la entrada: la id del grupo y de la sesión: el objeto del usuario.

```
public function store(Request $request) {
      $user = auth()->user();
      $groupId = $request->group;
      $user->groups()->attach($groupId);
      return redirect()->back(); }
```

5.5. Estudio de un contenido del curso

Por último, es importante protegerse cuando se intenta acceder a un contenido del curso sin estar inscrito en dicho curso. Para ello se crea el *middleware* `IsEnrolled`, que bloquea el paso, cuando se carga la ruta correspondiente, y ejecuta su código de comprobación. Sólo pueden tener acceso los usuarios que estén inscritos en el curso, luego se requiere dos parámetros: el id del curso (que se obtiene de la ruta: `$route->getParameter('course')`) y el usuario de la sesión (`auth()->user()`). En el modelo `User`, se crea el método:

```
public function isEnrolled($courseId)
{
    $result = \DB::select('
        select count(*)
        from groups as g
        join group_user as gu on g.id = gu.group_id
        join users as u on gu.user_id = u.id
        where g.course_id = ? and u.id = ?', [$courseId, $this->id]);
    return $result[0]->count != 0;
}
```

Mirando en la tabla pivote, se puede consultar si un estudiante está matriculado y desde la vista, se comprueba fácilmente: `@if(Auth::user()->isEnrolled($course->id)`, muy útil, por ejemplo, para ocultar el botón de inscripción.

Una vez finalizado el middleware, implementando su manejador, se registra en el archivo `Kernel.php` con el nombre `enrol` y ya se puede usar en el archivo de rutas.

```
Route::group([
  'middleware' => ['auth', 'enrol'],
  'namespace' => 'Course'],
  function () {
    Route::get('courses/{course}/{content}',
          ['as'=>'courses.content', 'uses'=>'ContentsController@code']);
    Route::post('courses/{course}/{content}',
          ['as'=>'courses.compile', 'uses'=>'ContentsController@compile']);
});
```

Cuando el alumno se ha inscrito en un curso, podrá comenzar a estudiar, en orden, un contenido del curso. Lo primero que hace el controlador es asociar dicho contenido al usuario, en una relación muchos a muchos, usando la tabla pivote `content_user`. Se mantendrá esta relación para registrar el comienzo del estudio y almacenar el número de errores, es decir, los intentos que el usuario ha necesitado para superar la lección. Después, se carga el contenido seleccionado y se lo devuelve a la vista encargada de organizarlo (`code.blade.php`).

Antes de continuar, la Tabla 9 muestra un breve resumen con las relaciones básicas que han sido necesarias para la implementación de Codeps.

Relación	***Modelo***	***Modelo (con la clave foránea)***
One To One	$this->hasOne('Model');	$this->belongsTo('Model');
One To Many	$this->hasMany('Model');	$this->belongsTo('Model');
Many To Many	$this->belongsToMany('Model);	$this->belongsToMany('Model);
Has Many Through	$this->hasManyThrough('ModelDestino', 'ModelIntermedio');	

Tabla 9. Relaciones en Eloquent

La vista se compone de tres partes principales. En la parte inicial se muestra la información del contenido: título, descripción (texto con la información teórica a estudiar), una imagen (opcional, si se desea complementar la teoría con alguna gráfica), el ejercicio a realizar por el usuario y un botón para mostrar un mensaje de ayuda.

En la siguiente parte, se inserta el área de texto donde el usuario podrá escribir su código de respuesta al ejercicio planteado. Se hace uso del *plugin* **Code Editor** de **CKEditor**, un editor con estilo personalizado, para decorar el área de texto. Más información sobre su instalación y configuración en [21].

Nota: el contenido que se guarde en el atributo `code` (que contiene el código inicial del ejercicio propuesto) debe estar encerrado entre las etiquetas **`<pre></pre>`**, para que el *plugin* pueda procesarlo y lo añada correctamente en el área de texto de HTML.

```
{!! Form::model($content, ['route' => 'courses.compile', 'method' => 'POST']) !!}
    {!! Form::textarea('code', null, ['class' => 'ckeditor']) !!}
    <br>
    <p class="text-center">
        <button type="submit" class="btn btn-info">
            Ejecutar
        </button>
    </p>
{!! Form::close() !!}
```

La última parte se corresponde cuando el usuario pulsa el botón de ejecutar. Se envía el formulario con el método POST y el controlador recibe el parámetro `code`, que contiene la respuesta del ejercicio, además del id del contenido, que se obtiene de la ruta.

El controlador se encarga de las siguientes tareas:

- Limpiar el código de posibles etiquetas que inserte HTML al enviar el formulario. Se usa la función de PHP: `$code = strip_tags(\Input::get('code'));`
- Crear un archivo con extensión .c en el directorio de almacenamiento temporal del proyecto `storage_path('app')` insertando el código con `fopen()` y `fwrite()`;
- **Compilación** del código: `exec('gcc -o a hello.c', $output, $return_value);` si el valor de retorno es distinto de cero se devuelve un mensaje con el error y se suma una unidad al valor del campo `errors` de la base de datos.
- **Ejecución** del programa: `exec('a.exe', $output);` se compara el resultado con el del campo `solution` de la base de datos y si coincide el ejercicio está correcto y en caso contrario, se devuelve un aviso, incrementándose también el valor `errors`.

Cuando se ha superado un contenido correctamente, el usuario puede continuar con el curso, seleccionando del menú deslizable el siguiente contenido del capítulo.

6. Pruebas y resultados

Además de las migraciones, se pueden usar *seeders* y Model Factories para una carga rápida y automática de información en la base de datos, con el objetivo de contar con bastantes datos para realizar las pruebas. Por ejemplo, cuando se ejecuta el siguiente seeder desde la línea de comandos: `php artisan db:seed --class=UserTableSeeder` se insertan 50 usuarios.

```
public function run()
{
    $faker = Faker::create();
    for ($i = 0; $i < 50; $i++)
    {
        $id = \DB::table('users')->insertGetId(array (
            'name'          => $faker->lastName,
            'email'         => $faker->unique()->email,
            'password'      => \Hash::make('123456'),
            'type'          => 'user'
        ));

        \DB::table('user_profiles')->insert(array (
            'user_id'       => $id,
            'biography'     => $faker->paragraph(rand(2, 5)),
            'website'       => 'http://www.' . $faker->domainName
        ));
    }
}
```

Para verificar el proceso, se usa la aplicación **phpPgAdmin**, que además permite realizar pruebas sobre las sentencias de consulta, de una manera intuitiva gracias a su sencilla interfaz.

Desarrollar sobre un patrón de diseño MVC facilita las pruebas de los métodos, pero a su vez, complica bastante el seguimiento del hilo de ejecución de la aplicación, requiriendo de herramientas más complejas. Se enumeran, por orden de complejidad, los métodos utilizados.

En el archivo `config/app.php` se puede activar el modo `debug`, muy útil en la etapa de desarrollo, para localizar los errores cuando se intenta cargar una ruta inválida en el navegador web, mostrándose todo el flujo que ha seguido la ejecución y las líneas de error.

Una medida de prevención muy útil a la hora de probar un controlador, es usar el *helper* de Laravel `dd($variable)`, parecido a la función `var_dump()` de PHP, que finaliza la ejecución de la aplicación e imprime el contenido de la variable que recibe como argumento.

En el transcurso de un entorno de desarrollo a uno de pruebas, es necesario llevar un registro de errores para continuar con el proceso de depuración de forma trasparente. **Laravel Log Viewer** es un paquete que registra todos los errores ocurridos en la aplicación, de gran utilidad cuando se deshabilita el modo `debug`.

Se ha instalado una barra de depuración **Laravel Debugbar**, que permite conocer rápidamente información de todo lo que se está ejecutando al cargar una página, como por ejemplo, se pueden obtener los nombres de una ruta asociada, las consultas SQL que se han ejecutado, la memoria cache utilizada o el tiempo de respuesta. Con este paquete se mejoran los problemas que afectan al rendimiento de Codeps, optimizando el código. Se puede consultar más información sobre su uso e instalación en [22].

En cuanto a pruebas unitarias, Laravel integra **PHPUnit** y **PHPSpec**. Estas herramientas permiten aislar cada parte de la funcionalidad de la aplicación y comprobar que funcionan individualmente. Se ha utilizado en el módulo de los cursos, a la hora de obtener resultados de las relaciones con sus contenidos. Para invocar una prueba creada con sus correspondientes `asserts`, que verifique el correcto funcionamiento de un método específico del controlador, que se encarga de obtener los cursos a través de un identificador, se ejecuta:

```
phpunit --bootstrap src/Course.php tests/CourseTest
```

Sin desviarse de la funcionalidad principal de la aplicación, uno de los objetivos de Codeps es informar tanto al profesor como al estudiante de los resultados obtenidos. Para ello se instalan algunos paquetes bastante útiles que permiten el trabajo con dichos datos y la elaboración de estadísticas para su posterior análisis. Tanto en el panel de administración como en la cuenta de un usuario, se integran algunas herramientas como DomPdf para generar documentos en PDF y Laravel Excel, que descarga los resultados en un archivo Excel [23].

7. Conclusiones

En este capítulo, se analizan las conclusiones finales del proyecto y se reflexiona sobre las posibles tareas de mantenimiento y extensión de la aplicación. Actualmente, anticiparse a lo que va a suceder o estar preparados para el cambio, forman las bases a tener en cuenta en cualquier aplicación web verdaderamente importante.

7.1. Conclusiones

Este trabajo ha consistido en el desarrollo de Codeps, una aplicación web para el aprendizaje autónomo de lenguajes de programación compilados como C. Las principales funcionalidades son las siguientes.

- Registro de estudiantes gracias a su módulo de usuarios.
- Administración de los recursos de la aplicación.
- Inscripciones de los usuarios en los cursos.
- Estudio de los contenidos de un curso, permitiendo la compilación en tiempo real, para facilitar la tarea al estudiante.
- Recogida de estadísticas y datos de interés que permitan a los profesores realizar análisis y valoraciones para mejorar la docencia.

Una de las novedades frente a otros sitios web es que esta aplicación permite compilar código sin necesidad de instalar entornos de programación auxiliares. Se busca adaptar el sitio al estudiante y que no se desvíe del objetivo final: el aprendizaje del curso.

Además, permite al profesor, como administrador del sistema, obtener estadísticas de los cursos, que le indiquen las posibles desventajas que puedan existir, para poder aplicar mejoras en caso necesario. Al mismo tiempo, provee de un mecanismo de seguimiento al usuario en particular, que estudia un curso, algo bastante útil para ver realmente su progreso.

La aplicación ha sido desarrollada con vistas a poder utilizarse en el primer curso de Grado de Ingeniería Informática, lo que tiene un atractivo añadido: crear una aplicación web que va a ser llevada a una fase de producción real y un mantenimiento posterior.

7.2. Trabajo futuro

Este entorno conlleva unas expectativas bastantes interesantes, ya que se ha diseñado para incorporar nuevos lenguajes, como Android, abiertos para otros cursos del Grado o incluso crear cursos especializados. Por ello, se llevará a cabo negociaciones con futuros proyectos que se comprometan a ser una extensión de esta aplicación, ya que se puede mejorar y aumentar las funcionalidades a la hora de analizar los datos obtenidos. Por ejemplo, además de conocer el número de intentos fallidos de un curso, se pueden hacer estudios más precisos sabiendo el número de horas dedicadas por un estudiante. Estas son algunas ideas de lo que se puede llegar a logar en futuras versiones.

La aplicación espera futuras actualizaciones con nuevas funcionalidades que aporten mayor posibilidad de aprendizaje, como por ejemplo, la inclusión de audio, videoconferencias o video tutoriales a un contenido teórico.

También conviene realizar un esfuerzo adicional en el aspecto del rendimiento, creando un sitio más seguro y usando componentes dinámicos junto con procesos asíncronos con AJAX, para no recargar las vistas. En cuanto a seguridad, el *framework* estudiado ya cuenta con algunos mecanismos como las validaciones de los formularios, para protegerse de CSRF y las inyecciones SQL, pero se seguirá mejorando este aspecto tan importante. Prueba de ello, es que Laravel se ha actualizado a la versión LTS 5.1 y recibirá soporte a largo plazo, por lo que ya se está trabajando en la siguiente versión de Codeps.

8. Referencias

[1] Wikipedia, la enciclopedia libre, «Aprendizaje electrónico,» [En línea]. Available: https://es.wikipedia.org/wiki/Aprendizaje_electr%C3%B3nico.

[2] Recursos educ.ar, «Cómo evaluar sitios web,» [En línea]. Available: http://www.educ.ar/recursos/ver?id=92759.

[3] J. Cerezo, «Sé 'responsive'. Tecnología. El País,» 21 Abril 2005. [En línea]. Available: http://tecnologia.elpais.com/tecnologia/2015/04/20/actualidad/1429561823_371980.html.

[4] Wikipedia, la enciclopedia libre, «Computación en la nube,» [En línea]. Available: https://es.wikipedia.org/wiki/Computaci%C3%B3n_en_la_nube.

[5] J. F. Kurose y K. W. Ross, Redes de computadoras. Un enfoque descendente, 5 ed., Pearson, 2010.

[6] R. S. Pressman, Ingeniería del software. Un enfoque práctico, 5 ed., McGraw Hill, 2002.

[7] I. Sommerville, Ingeniería de Software, 7 ed., Addison Wesley, 2005.

[8] PHP.net, «PHP: Espacios de nombres - Manual,» [En línea]. Available: http://php.net/manual/es/language.namespaces.php.

[9] Wikipedia, la enciclopedia libre, «Cross-site scripting,» [En línea]. Available: http://es.wikipedia.org/wiki/Cross-site_scripting.

[10] M. Pratt, «Patrones de Diseño: Inyección de Dependencias,» 16 Abril 2013. [En línea]. Available: http://www.michael-pratt.com/blog/13/Patrones-de-Diseno-Inyeccion-de-Dependencias.

[11] R. Elmasri y S. B. Navathe, Fundamentos de Sistemas de Bases de Datos, 5 ed., Addison Wesley, 2007.

[12] Wikipedia, la enciclopedia libre, «Inyección SQL,» [En línea]. Available: https://es.wikipedia.org/wiki/Inyecci%C3%B3n_SQL.

[13] M. Stow, «What's my Viewport Size?,» 2013. [En línea]. Available: http://viewportsizes.com/mine.

[14] Lipis, «Flag-icon-css,» [En línea]. Available: https://github.com/lipis/flag-icon-css.

[15] Mydnic, «Laravel 5 And His F*cking non-persistent App SetLocale,» 17 Marzo 2015. [En línea]. Available: http://mydnic.be/post/laravel-5-and-his-fcking-non-persistent-app-setlocale.

[16] A. Engebretson, «Laravel Collective,» 2015. [En línea]. Available: http://laravelcollective.com/docs/5.1/html.

[17] T. Otwell, «Laravel - The PHP Framework For Web Artisans,» [En línea]. Available: http://laravel.com/docs/5.1.

[18] Laravel, «Laravel API,» 2015. [En línea]. Available: http://laravel.com/api/5.1.

[19] Codetutorial, «Laravel 5 File upload, storage and download,» 8 Febrero 2015. [En línea]. Available: http://www.codetutorial.io/laravel-5-file-upload-storage-download.

[20] J. Ochoa, «Cómo subir archivos con Laravel 5,» 15 Abril 2015. [En línea]. Available: https://styde.net/sistema-de-archivos-y-almacenamiento-en-laravel-5.

[21] P. Baron, «Code Editor | CKEditor.com,» [En línea]. Available: http://ckeditor.com/addon/pbckcode.

[22] B. Heuvel, «Laravel-debugbar - Packagist,» [En línea]. Available: https://packagist.org/packages/barryvdh/laravel-debugbar.

[23] Maatwebsite Team, «Laravel Excel Documentation,» [En línea]. Available: http://www.maatwebsite.nl/laravel-excel/docs.

[24] J. Eguiluz, Introducción a AJAX, Librosweb.

[25] Wikipedia, la enciclopedia libre, «Codecademy,» [En línea]. Available: https://es.wikipedia.org/wiki/Codecademy.

[26] Wikipedia, la enciclopedia libre, «Modelo-vista-controlador,» [En línea]. Available: https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador.

[27] W. J. Gilmore, Easy Laravel 5, easylaravelbook.com.

[28] D. Rees, Laravel: Code Bright, Leanpub, 2013.

[29] Laravelbook, «Architecture of Laravel Applications,» [En línea]. Available: http://laravelbook.com/laravel-architecture/.

[30] C. Álvarez, «Patrones de Diseño (Active Record vs DAO),» 31 Julio 2014. [En línea]. Available: http://www.genbetadev.com/java-j2ee/patrones-de-diseno-active-record-vs-dao.

[31] w3schools.com, «HTML Reference,» [En línea]. Available: http://www.w3schools.com/tags/default.asp.

Glosario

AJAX: Asynchronous JavaScript and XML. Tecnología que permite la comunicación entre el servidor y el cliente de forma asíncrona.

CMS: Content Management System. Sistema de gestión de contenido, que permite la creación y administración de contenidos, que principalmente se utiliza en páginas web.

Cookies: archivos de texto de tamaño mínimo, que se almacenan en los clientes de las páginas webs mediante sus navegadores Web, por petición de los servidores Web visitados. Permite guardar información como nombres, contraseñas o preferencias de la página, del cliente, que el servidor recuperará e interpretará de manera automática en sucesivas visitas.

CSS: Cascading Style Sheets. Lenguaje de formato que permite definir los estilos de una página web. El World Wide Web Consortium (W3C) es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los navegadores.

Framework: estructura de soporte definida, en la cual un proyecto de software puede ser organizado y desarrollado en una plataforma o marco de trabajo.

HTML: Hyper Text Markup Language. Lenguaje de marcado de contenidos en páginas web basado en etiquetas.

HTTP: HyperText Transfer Protocol. Protocolo de intercambio de información vía web.

IDE: Entorno de Desarrollo Integrado. Aplicación compuesta por un conjunto de herramientas útiles para la programación.

JavaScript: lenguaje de programación interpretado que se utiliza para crear páginas web dinámicas, es decir, que incorporan efectos y/o acciones.

JQuery: librería JavaScript sencilla, eficiente y con multitud de funcionalidades que permite, entre otras cosas, manipular el código HTML de forma dinámica (en tiempo de ejecución).

JSON: JavaScript Object Notation. Formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

LTS: Long Term Support. Término usado para nombrar versiones especiales de software de código abierto, diseñadas para tener soportes durante un periodo más largo.

Middleware: software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, o paquetes de programas, redes, hardware y/o sistemas operativos.

MVC: Modelo Vista Controlador. Patrón de arquitectura de software, utilizado en aplicaciones Web, encargado de separar la lógica de negocio de la interfaz del usuario.

PSR-4: conjunto de especificaciones creadas por el grupo de interoperabilidad de PHP (http://www.php-fig.org). Es totalmente recomendable seguir este protocolo para estructurar todas las clases de un proyecto en Laravel.

REST: Representational State Transfer. Es un estilo de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web. Describe la interfaz entre sistemas que utilice directamente HTTP para obtener datos o ejecutar operaciones bien definidas, sobre ellos, en cualquier formato (XML, JSON, etc) sin las abstracciones adicionales de los protocolos basados en el intercambio de mensajes, como por ejemplo SOAP.

SGBD: Sistema Gestor de Base de Datos. Programa que permite definir, crear y utilizar una base de datos.

SQL: Structured Query Language. Lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre los datos.

SQL injection: Vulnerabilidad consistente en la inserción de una sentencia SQL dentro de otra sentencia SQL a su vez, por la cual un usuario malintencionado puede leer o modificar cualquier dato contenido en la Base de Datos.

SSL: Secure Sockets Layer (capa de conexión segura). Protocolo criptográfico que proporciona comunicaciones seguras por una red, comúnmente Internet.

TAW: Test de Accesibilidad Web, consiste en una herramienta desarrollada por CTIC para el análisis de la accesibilidad de los sitios web, alcanzando de una manera global e integral a todos los elementos y páginas que componen dicho sitio. Consiste en permitir el acceso a todas las personas a ese sitio, independientemente de sus características diferenciadoras.

URL: Uniform Resource Locator. Localizador de recursos uniforme. Secuencia de caracteres, de acuerdo a un formato estándar, que se usa para nombrar recursos en Internet para su localización o identificación.

VPN: Virtual Private Network (red privada virtual). Tecnología de red que permite una extensión segura de la red local sobre una red pública o no controlada.

W3C: Word Wide Web Consortium. Consorcio internacional productor de estándares para los navegadores web. Estos estándares pasan por varios estados hasta que llegan a su aprobación y consecuente recomendación.

WWW: World Wide Web. Sistema de distribución de información basado en hipertextos enlazados y accesibles a través de Internet.

XML: Extensible Markup Language. Es un lenguaje de marcas desarrollado por W3C para almacenar datos de forma legible. A diferencia de otros lenguajes, da soporte a bases de datos, siendo útil cuando varias aplicaciones deben comunicarse entre sí o integrar información.

Anexos

Anexo A: Instalación de Apache 2.4.12 en Windows 8.1

Descargar el archivo ***httpd-2.4.12-win64-VC11.zip*** de la página web oficial para Windows (http://www.apachelounge.com/download) y descomprimirlo en el directorio:

```
D:\Servidor\apache
```

Configurar el servidor Apache desde el archivo: D:\Servidor\apache\conf\httpd.conf

Hay que modificar las siguientes líneas:

```
ServerRoot "D:\Servidor\apache"
Listen 80
Listen 8080
```

Se debe añadir los siguientes módulos:

```
LoadModule rewrite_module modules/mod_rewrite.so
LoadFile "D:\Servidor\php\php5ts.dll"
LoadModule php5_module "D:\Servidor\php\php5apache2_4.dll"
AddHandler application/x-httpd-php .php
```

Hay que indicar el directorio raíz y algunos datos más:

```
PHPIniDir "D:\Servidor\php"
ServerAdmin admin@example.com
ServerName 127.0.0.1:8080
DocumentRoot "D:\Servidor\www"
<Directory "D:\Servidor\www">
AllowOverride All
</Directory>
ScriptAlias /cgi-bin/ "D:\servidor\apache\cgi-bin\"
<Directory "D:\Servidor\apache\cgi-bin">
DirectoryIndex index.php index.phtml index.html
```

Crear la variable de entorno APACHE y añadirla al path:

```
Path        %APACHE%\bin;…
APACHE      D:\Servidor\apache
```

Instalar el servidor e iniciarlo manualmente (se puede verificar la versión instalada):

```
httpd -k install
httpd -k start
httpd –v
```

Anexo B: Instalación de PHP 5.6.7 en Windows 8.1

Descargar el archivo ***php-5.6.7-Win32-VC11-x64.zip*** con Thread Safe de la página oficial para Windows (http://windows.php.net/download) y descomprimirlo en el directorio:

```
D:\Servidor\php
```

Configurar PHP para habilitar algunas extensiones necesarias para la aplicación. Para ello hay que renombrar el archivo el php.ini-development por D:\Servidor\php\php.ini

```
default_charset = "UTF-8"
extension_dir = "D:\Servidor\php\ext"
extension=php_bz2.dll
extension=php_curl.dll
extension=php_fileinfo.dll
extension=php_gd2.dll
extension=php_gmp.dll
extension=php_intl.dll
extension=php_mbstring.dll
extension=php_mysqli.dll
extension=php_openssl.dll
extension=php_pdo_mysql.dll
extension=php_soap.dll
extension=php_sockets.dll
extension=php_tidy.dll
extension=php_xmlrpc.dll
extension=php_xsl.dll
date.timezone ="Europe/Madrid"
SMTP = smtp.tusitio.com
smtp_port = 25
sendmail_from = tucorreo@correo.com
```

Crear la variable de entorno APACHE y añadirla al path:

```
Path        %PHP%; …
PHP         D:\Servidor\php
```

Se puede verificar que la versión ha sido instalada correctamente desde línea de comandos:

```
php -v
```

Se puede instalar el IDE de desarrollo para PHP ***phpStorm-8.0.3.exe*** de la página oficial (https://www.jetbrains.com/phpstorm) en el directorio: D:\Servidor\phpstorm

Anexo C: Instalación de PostgreSQL 9.4.1 en Windows 8.1

Descargar el archivo ***postgresql-9.4.1-3-windows-x64.exe*** de la página oficial para Windows (http://www.postgresql.org/download/windows) e instalarlo en el directorio:

```
D:\Servidor\pgsql
```

Se puede configurar en los siguientes archivos de configuración:

```
D:\Servidor\pgsql\data\postgresql.conf
D:\Servidor\pgsql\data\pg_hba.conf
```

Añadir las siguientes variables de entorno para que a la hora de utilizar PostgreSQL desde línea de comandos, no sea necesario indicar estos datos:

```
Path                %POSTGRE%\bin; …
POSTGRE             D:\Servidor\pgsql
PGDATA              D:\Servidor\pgsql\data
PGDATABASE          codeps
PGLOCALEDIR         D:\Servidor\pgsql\share\locale
PGPORT              5432
PGUSER              postgres
```

Para verificar la versión instalada y que todo está funcionando correctamente:

```
psql -V
pg_ctl status
```

Adicionalmente se puede descargar la aplicación ***PhpPgAdmin-5.1.zip*** desde su página oficial (http://phppgadmin.sourceforge.net/doku.php?id=download) y descomprimirlo en:

```
D:\Servidor\www
```

Se puede configurar en el archivo D:\Servidor\www\phpPgAdmin-5.1\conf\config.inc.php

```
$conf['servers'][0]['host'] = 'localhost';
$conf['extra_login_security'] = false;
```

Anexo D: Entorno de desarrollo con Laravel 5 en Windows 8.1

Para poder consultar aplicaciones creadas con PHP para Windows 8, se necesita seguir los anexos anteriores para instalar y configurar un entorno de trabajo. Esto sólo es una forma de las varias posibilidades que existe, pero con un objetivo en común: poder tener acceso, desde toda la red local, con estas direcciones, desde el navegador Web:

```
http://127.0.0.1:8080
http://localhost:8080
```

Fíjese que para tener acceso remoto desde Internet (fuera de la red local) hay que configurar las opciones de reenvío del router, indicando IP y puertos, internos y externos, y acceder con:

```
http://IPpública:PuertoExterno
```

Para hacer uso del framework **Laravel 5**, hay que ir un paso más allá y configurar un host virtual, igualmente, todo ello en local, ya que se supone que no se ha contratado ningún dominio Web que incluya un servidor DNS que resuelva las direcciones y un servidor Web donde alojar los datos de la aplicación Web.

También hay que indicar que existen otras alternativas, como **Homestead**. Laravel nos proporciona un entorno completo más profesional para nuestros proyectos. No hay que preocuparse en configurar un servidor ni cualquier otra herramienta. Desde la documentación de la página oficial, se explica cómo instalar una máquina virtual (VMWare o VirtualBox) y **Vagrant**, junto con todo el software que ya viene incluido. Este paquete nos permite tener una mayor facilidad de instalación y configuración.

Dado que es un proyecto individual, no tiene mucho sentido instalar Vagrant en una máquina virtual y conectarse por ssh, pero cuando se quiere tener un entorno de producción seguro con varios desarrolladores trabajando a la vez en un mismo proyecto, puede ser una buena opción.

En este caso, se opta por Laravel 5 en Windows 8 con Apache, PHP y PostgreSQL. Para crear una aplicación, se instala el manejador de dependencias **Composer** y desde la unidad D:

```
cd "Servidor\www"
composer create-project laravel/laravel codeps --prefer-dist
```

Se puede acceder a la página principal de la aplicación usando:

```
http://localhost:8080/codeps/public
```

Para añadir al servidor un host virtual en local, primero se tiene que añadir en el archivo C:\Windows\System32\drivers\etc\hosts:

```
127.0.0.1    localhost
127.0.0.1    codeps.es
```

A continuación se tiene que habilitar en D:\Servidor\apache\conf\httpd.conf:

```
Include conf/extra/httpd-vhosts.conf
```

Luego hay que crear el host virtual, donde se indica el puerto 80 para que lo cargue por defecto, el nombre de la aplicación y la carpeta que será de acceso público a través del navegador, asegurándose dejar en el puerto 8080 la dirección raíz.

En D:\Servidor\apache\conf\extra\httpd-vhosts.conf:

```
<VirtualHost *:80>
	ServerAdmin webmaster@codeps.es
	DocumentRoot "D:\Servidor\www\codeps\public"
	ServerName codeps.es
	ServerAlias www.codeps.es
	ErrorLog "logs/codeps.es-error.log"
	CustomLog "logs/codeps.es-access.log" common
</VirtualHost>

<VirtualHost *:8080>
	ServerAdmin admin@example.com
	DocumentRoot "D:\Servidor\www"
	ServerName localhost
	ServerAlias localhost
	ErrorLog "logs/error.log"
	CustomLog "logs/access.log" common
</VirtualHost>
```

Una vez reiniciado el servidor, se puede comprobar que se puede acceder correctamente:

```
httpd -k restart
http://codeps.es
```

A partir de ahora, cualquier cambio en la aplicación, se debe usar Artisan en la consola, desde el directorio del proyecto actual. Para instalar otro framework como Bootstrap o jQuery, simplemente hay que descomprimir los archivos en la carpeta `public` de la aplicación.

Anexo E: Manual de comandos de Artisan CLI

Artisan es el nombre de la interfaz de línea de comandos incluida en Laravel. A continuación se enumeran algunos de los comandos útiles para el desarrollo de la aplicación. Se debe estar dentro del directorio del proyecto.

`php artisan list` (Listado con todos los comandos disponibles)

`php artisan help comando` (Ayuda del comando especificado)

`php artisan route:list` (Listado todas las rutas registradas en la aplicación)

`composer update` (Cuando se añade un nuevo componente en composer.json)

`composer dump-autoload` (Cuando se añade una nueva migración o seeder)

`php artisan migrate:install` (Instala el sistema de migraciones)

```
php artisan make:migration create_nombre_table --create="nombre"
```

```
php artisan migrate:rollback
```

```
php artisan migrate:reset
```

`php artisan migrate` (Realiza las migraciones que faltan por hacer)

`php artisan migrate:refresh` (Realiza un reset y luego un migrate)

`php artisan db:seed` (Ejecuta todos los seeders)

`php artisan db:seed --class=NombreTableSeeder` (Ejecuta el seeder especificado)

`php artisan migrate:refresh --seed` (Realiza la migración y luego los seeders)

```
php artisan make:model NombreModelo
```

```
php artisan make:controller NombreController
```

```
php artisan make:request NombreRequest
```

```
php artisan make:middleware Nombre
```

Anexo F: Manual de usuario de Codeps

La aplicación Codeps está formada por una serie de vistas, donde a continuación, se muestra alguna de ellas. El usuario puede acceder desde el servidor local en el virtual host creado en Apache (http://codeps.es) o conectándose a su IP pública. Estas vistas pueden variar en la actualidad, dependiendo del mantenimiento llevado a cabo tras la entrega de la primera versión del proyecto.

En la Figura 20 se puede observar la vista principal de la aplicación, cuando el usuario se registra e inicia sesión correctamente. En la cabecera, el usuario tiene disponible un enlace al módulo de los cursos y el menú con acceso al perfil o al panel de administración, en el caso de que sea administrador.

Figura 20. Vista "home" de Codeps

La Figura 21 muestra una versión de la misma vista principal, pero adaptable para dispositivos móviles, disminuyendo el ancho y alto de la pantalla. En ella se puede apreciar como el menú se transforma en un botón deslizable y los elementos del cuerpo se ordenan.

Figura 21. Vista "home" adaptable

Dos de las tareas más críticas de la aplicación son: la inscripción de un usuario en uno de los grupos disponibles de un curso y la ejecución del código cuando se estudia un contenido.

La vista de la primera de ellas viene dada por la Figura 22. La pantalla modal con la selección de grupos se pone en primer plano, sobre la vista general del detalle de un curso (pinchando sobre la imagen en la vista anterior con el listado de todos los cursos).

La Figura 23 representa el detalle de un contenido, cuando en la vista anterior se selecciona desde un capítulo. El botón "ejecutar" procesa la respuesta del usuario, mostrando su mensaje correspondiente.

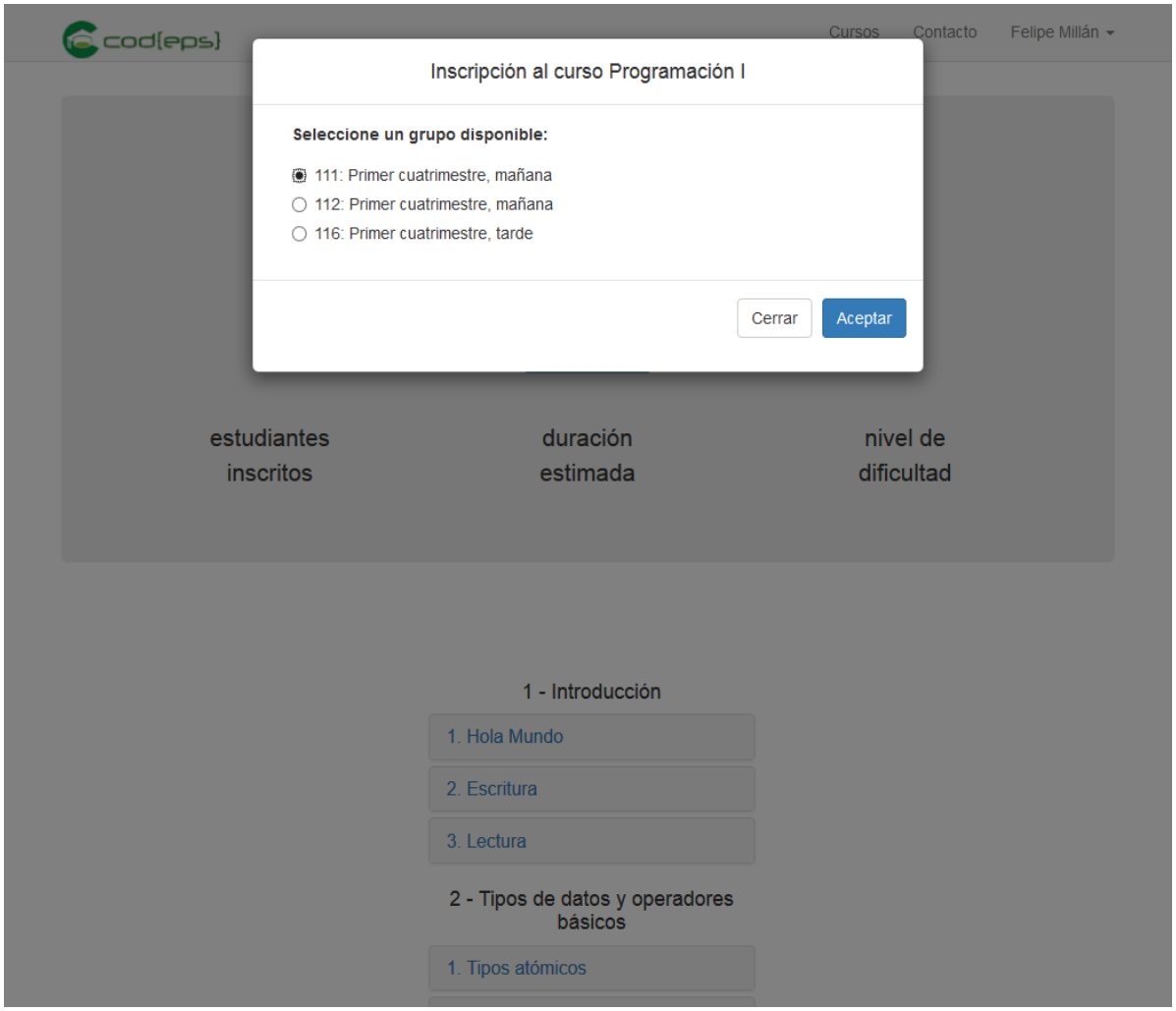

Figura 22. Vista de inscripciones de Codeps

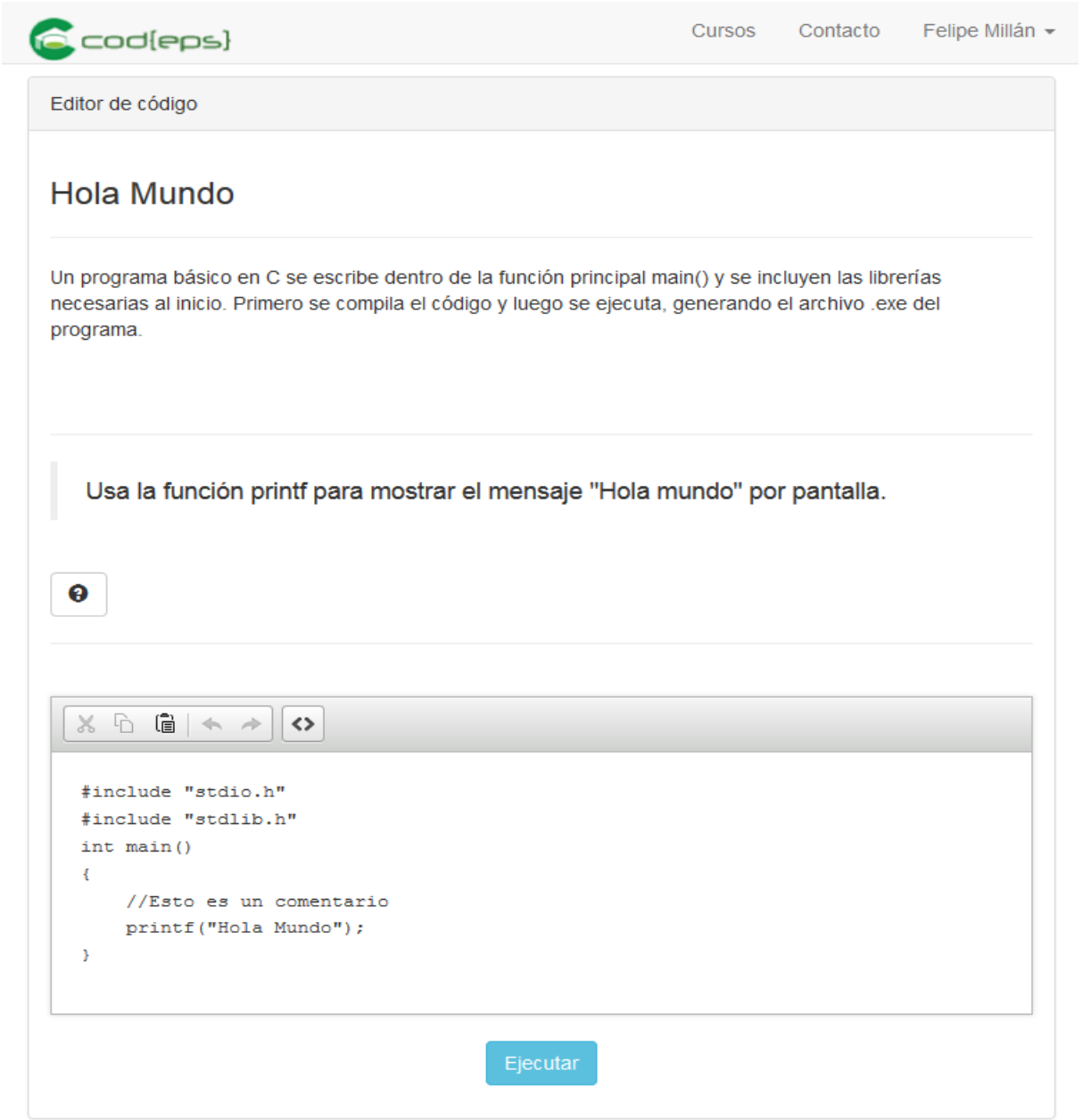

Figura 23. Vista de un contenido de Codeps